

CHI Systems, Inc.
100 Burns Place
Goleta, CA 93117
(805) 984-8868

AD A129383

ACOUSTICAL ARRAY PROCESSOR
DESIGN

Final Technical Report

June 1983

PRINCIPAL INVESTIGATOR: Dr. Glen J. Culler

PROJECT SCIENTISTS: Dr. Judith B. Bruckner
Dr. Michael McCammon

This research was supported
by the Defense Advanced Re-
search Projects Agency un-
der ARPA Order No. 3625.
Contract MDA 903-78-C-0313.

Distribution of this document
is unlimited. It may be re-
leased to the Clearinghouse,
Department of Commerce for
sale to the general public.

The views and conclusions contained in this document are those
of the authors and should not be interpreted as necessarily
representing the official policies, either expressed or
implied, of the Defense Advanced Research Projects Agency or
the U.S. Government.

DTIC
SELECTED
JUN 16 1983
A

DTIC FILE COPY

83 06 16 014

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER MDA 903-78-C-0313-F	2. GOVT ACCESSION NO. AD-A129	3. RECIPIENT'S CATALOG NUMBER 383
4. TITLE (and Subtitle) Accoustical Array Processor Design		5. TYPE OF REPORT & PERIOD COVERED Final Technical Report
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) J/B. Bruckner M. McCammon		8. CONTRACT OR GRANT NUMBER(s) MDA 903-78-C-0313
9. PERFORMING ORGANIZATION NAME AND ADDRESS CHI Systems, Inc. 100 Edward Burns Place Goleta, CA 93117		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS ARPA ORDER NO. 3625
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Supply Service - Washington Room 1D245, The Pentagon Washington, D.C. 20310 (T. Bushnell)		12. REPORT DATE June 8, 1983
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) DCASMA Van Nuys (S0512A) 6230 Van Nuys Blvd. Van Nuys, CA 91408		13. NUMBER OF PAGES 57
		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Distribution of this document is unlimited. It may be released to the general public.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES Chief Scientist: Glen J. Culler Effective Date of contract-June 12, 1978 Expiration Date of contract-February 15, 1982		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Accoustical Array Processor, Array Processor LPC Voice Terminal, Logical Design		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) This report describes the design of the accoustical array processor known as the CHI-5. The primary goal of the architecture was to provide hardware efficient, yet economical, in implementing linear predictive coding (LPC) algorithms for analysis and synthesis of speech. The CHI-5 can be used with a host processor or it can be used as a voice terminal in a stand alone data driven mode.		

DD FORM 1 JAN 73 1473

EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

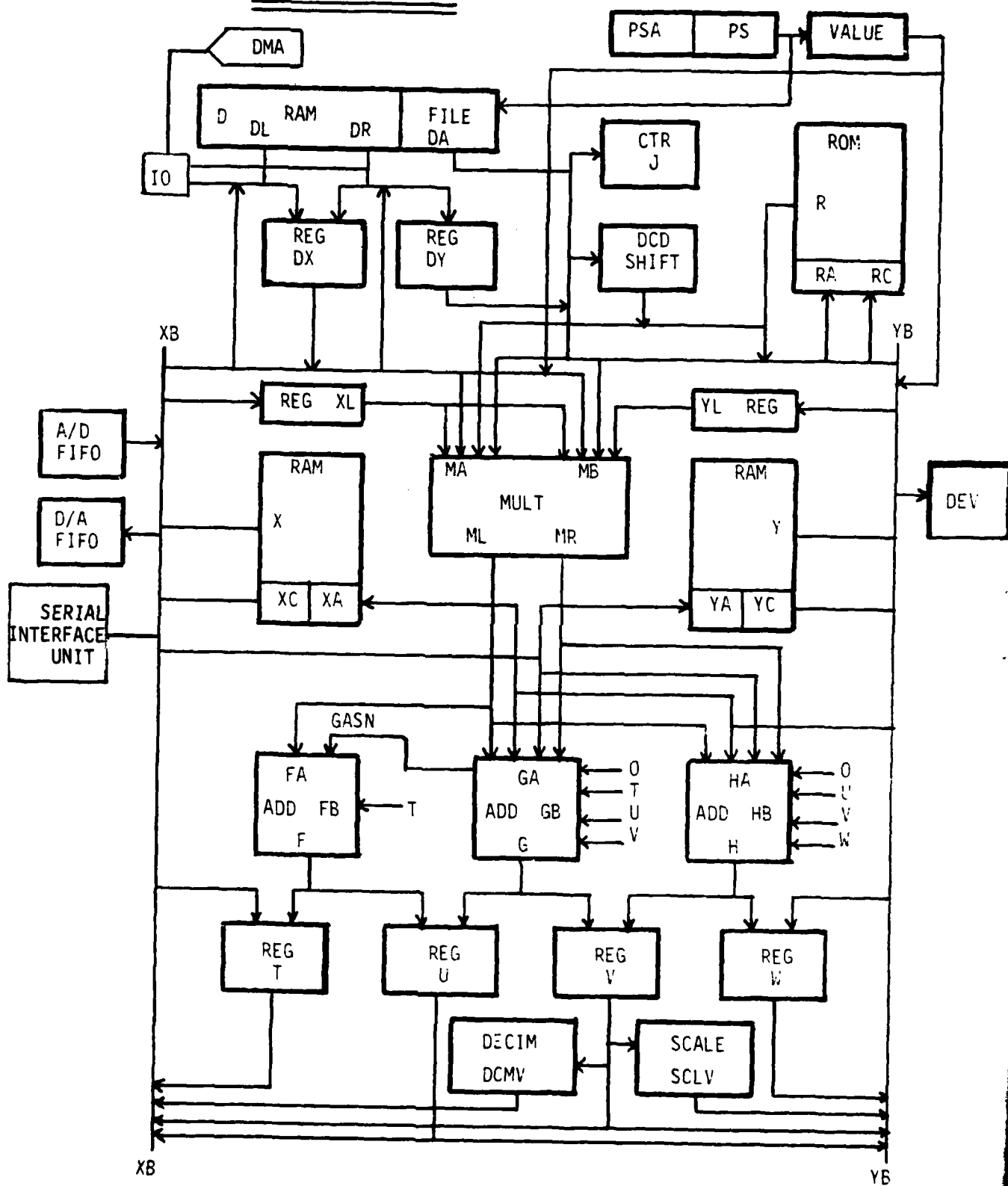
In J. Makhoul's paper "Stable and Efficient Lattice Methods for Linear Prediction", the best known methods of LPC analysis are brought together in a common framework. We chose this as our principal reference, partly because of its clarity, but mostly because it reveals a wide range of computational complexities engaged in the LPC techniques. For the requirements of the LPC algorithms, we determined that by providing an exact double precision dot product of vectors of up to 256 coordinates, and providing upper and lower bound sizing hardware, signals of 12-bit dynamic range can successfully be processed by LPC analysis and synthesis, using a 12-bit data element.

The contract, MDA903-78-C-0313, was later expanded to extend the design of the LPCAP to a processor which could be used as a stand-alone LPC voice terminal, or, with a host computer, as a research tool in LPC applications. The contract called for delivery of five of these processors. Now, the ARPA researchers that may be expected to use these devices have applications requiring greater numerical precision. As a result, we designed a 16-bit version of the LPCAP with a more general and less minimal framework. This processor is called the CHI-5.

SECRET

1. Section 101
 2. Section 102
 3. Section 103
 4. Section 104
 5. Section 105
 6. Section 106
 7. Section 107
 8. Section 108
 9. Section 109
 10. Section 110
 11. Section 111
 12. Section 112
 13. Section 113
 14. Section 114
 15. Section 115
 16. Section 116
 17. Section 117
 18. Section 118
 19. Section 119
 20. Section 120
 21. Section 121
 22. Section 122
 23. Section 123
 24. Section 124
 25. Section 125
 26. Section 126
 27. Section 127
 28. Section 128
 29. Section 129
 30. Section 130
 31. Section 131
 32. Section 132
 33. Section 133
 34. Section 134
 35. Section 135
 36. Section 136
 37. Section 137
 38. Section 138
 39. Section 139
 40. Section 140
 41. Section 141
 42. Section 142
 43. Section 143
 44. Section 144
 45. Section 145
 46. Section 146
 47. Section 147
 48. Section 148
 49. Section 149
 50. Section 150
 51. Section 151
 52. Section 152
 53. Section 153
 54. Section 154
 55. Section 155
 56. Section 156
 57. Section 157
 58. Section 158
 59. Section 159
 60. Section 160
 61. Section 161
 62. Section 162
 63. Section 163
 64. Section 164
 65. Section 165
 66. Section 166
 67. Section 167
 68. Section 168
 69. Section 169
 70. Section 170
 71. Section 171
 72. Section 172
 73. Section 173
 74. Section 174
 75. Section 175
 76. Section 176
 77. Section 177
 78. Section 178
 79. Section 179
 80. Section 180
 81. Section 181
 82. Section 182
 83. Section 183
 84. Section 184
 85. Section 185
 86. Section 186
 87. Section 187
 88. Section 188
 89. Section 189
 90. Section 190
 91. Section 191
 92. Section 192
 93. Section 193
 94. Section 194
 95. Section 195
 96. Section 196
 97. Section 197
 98. Section 198
 99. Section 199
 100. Section 200
 101. Section 201
 102. Section 202
 103. Section 203
 104. Section 204
 105. Section 205
 106. Section 206
 107. Section 207
 108. Section 208
 109. Section 209
 110. Section 210
 111. Section 211
 112. Section 212
 113. Section 213
 114. Section 214
 115. Section 215
 116. Section 216
 117. Section 217
 118. Section 218
 119. Section 219
 120. Section 220
 121. Section 221
 122. Section 222
 123. Section 223
 124. Section 224
 125. Section 225
 126. Section 226
 127. Section 227
 128. Section 228
 129. Section 229
 130. Section 230
 131. Section 231
 132. Section 232
 133. Section 233
 134. Section 234
 135. Section 235
 136. Section 236
 137. Section 237
 138. Section 238
 139. Section 239
 140. Section 240
 141. Section 241
 142. Section 242
 143. Section 243
 144. Section 244
 145. Section 245
 146. Section 246
 147. Section 247
 148. Section 248
 149. Section 249
 150. Section 250
 151. Section 251
 152. Section 252
 153. Section 253
 154. Section 254
 155. Section 255
 156. Section 256
 157. Section 257
 158. Section 258
 159. Section 259
 160. Section 260
 161. Section 261
 162. Section 262
 163. Section 263
 164. Section 264
 165. Section 265
 166. Section 266
 167. Section 267
 168. Section 268
 169. Section 269
 170. Section 270
 171. Section 271
 172. Section 272
 173. Section 273
 174. Section 274
 175. Section 275
 176. Section 276
 177. Section 277
 178. Section 278
 179. Section 279
 180. Section 280
 181. Section 281
 182. Section 282
 183. Section 283
 184. Section 284
 185. Section 285
 186. Section 286
 187. Section 287
 188. Section 288
 189. Section 289
 190. Section 290
 191. Section 291
 192. Section 292
 193. Section 293
 194. Section 294
 195. Section 295
 196. Section 296
 197. Section 297
 198. Section 298
 199. Section 299
 200. Section 300
 201. Section 301
 202. Section 302
 203. Section 303
 204. Section 304
 205. Section 305
 206. Section 306
 207. Section 307
 208. Section 308
 209. Section 309
 210. Section 310

FIGURE 1. CHI-5 ARCHITECTURE



The CHI-5 architecture is covered by U.S. patent #4,287,566

1. SYSTEM DESCRIPTION

Figure 1 shows the detailed structure of the CHI-5. Five separate memory banks are used: Data Left (DL) memory, Data Right (DR) memory, Arithmetic Processor Unit (APU) X memory (X), APU Y memory (Y), and Program (P) memory. The program controller consists of the P memory sequencer and macro-code decoder logic. The APU consists of a multiplier, three data adders, four accumulator registers, two address controllers and special logic functions for floating point scaling, and for data decimation. The DX and DY registers are memory output registers connecting the DL/DR busses to the X and Y busses. Bus drivers allow 16-bit input data from an external source to enter either DL or DR memory depending on the DA address. Output data from the DL bus/DR bus is sent to an external destination. The FIFOs on the X bus are special I/O memories for the analog-to-digital and digital-to-analog converters used in speech applications.

The DL and DR memories are used for data storage, for macro program storage, for micro-code download buffering and for I/O buffers. Each D memory module provides 8K double words of RAM and 1K or 2K of ROM.

Each X and Y array memory is a 1K word RAM. They can be accessed simultaneously and independently by the APU. These RAMs provide large scratch memory for the APU as well as buffers for data to be processed. The data can be transferred to the X/Y memories from the D memories or from the special I/O FIFOs.

The P memory contains the executable micro-code and is implemented as an 80-bit wide by 3K deep memory. 2K of this memory is ROM and the remaining 1K is RAM. The loading of P memory RAM is a special mode in which micro-code stored in D memory is then transferred into the program memory RAMs for eventual execution. This mode not only allows for micro-code checkout but also permits the CHI-5 to be reprogrammed for a variety of users or applications.

The P memory sequencer provides the controls and addressing capability for the microprogram memory. The sequencer performs sequential accesses to P memory as well as conditional or unconditional branches. This unit also provides for micro-code subroutines with a 16 deep stack file. A loop counter is also included in the sequencer for ease of repeating a sequence of micro-code. Interrupt signals from I/O devices cause an automatic branch to an interrupt routine.

The D memory controller provides the addressing capability for the D memories. A file of 16 D memory address registers is provided and can be loaded or read by the CHI-5. Eight registers are used for externally-controlled I/O operations. The remaining eight registers are used for array processor functions, such as macro-instruction program counter, macro-program stack pointer and data addressing.

The macro-code decoder logic allows data in the DL memory to be interpreted as two micro-code address fields. The first 5-bit field (MAC1) will be used as the address in program ROM of the first macro-instruction. The second 11-bit field (MAC2) is the address of the body of the microprogram which implements the macro. The decoder also recognizes certain codes that perform special hardware functions, such as using the contents of the DY register as the D memory address.

The 80-bit micro-code word is divided up into a number of fields, each of which is associated with a particular set of hardware elements. Figure 2 shows the field assignments of the micro-code instruction word for the system. Figure 3 indicates the micro-operations that can be performed with the CHI-5. Explanations of the symbols used for module, field, and operation names appear in the glossary which follows section 2.

The CHI-5 contains sufficient ROM in the P memory (PH), so that when reset, the P memory has a bootstrap program that can control the system enough to use the Host Interface to load micro-code into the P memory RAM (PS) and begin normal execution when the Host signals the transfer is complete. In a stand alone voice terminal application, the ROM program and data memories contain all of the programs necessary for the limited application. In this case, RESET starts the voice processing algorithms and the program continues to run without intervention by a Host.

FIGURE 2
MICROCODE FIELD ASSIGNMENTS
(See Glossary For Field Descriptions)

FIELD	Bit positions in octal (bit 0 is on right)
P	117-114
TEST	113-112
R	111
RA	110
M	107-105
MULT	104-100
D	103-100
ID	103-100
HCI	102
RLD	101-100
F	77-75
FG	74
G	73-70
XB	67-65
GB	64-63
TU	62-60
XA	57-54
X	53-50
VALUE	47-30
GH	27
HA	26-25
H	24-22
XL	21
YL	20
YB	17-15
HB	14-13
VW	12-10
YA	7-4
Y	3-0

Note: The following fields overlap wholly or in part: MULT, D, ID, HCI, RLD.

FIGURE 3. CU1-5 MICRO OPERATIONS CHART

CODE	PSA	XA	X	YA	Y	G	D
0	SEQ	NOOP	NOOP	NOOP	NOOP	0	READ
1	IF STAT=0	INC XA	XB-X	INC YA	YB-Y	GB-ML	WRITE
2	IF STAT=1	INC XA(2)	XB=X	INC YA(2)	YB=Y	ML-GB	READ(#)
3	HA	DEC XA	XB-XC	DEC YA	YB-YC	ML+GB	WRITE(#)
4	IF EQ	DEC XA(2)	XB=XC	DEC YA(2)	YB=YC	XB +GB	READ, INC DA(YB)
5	IF LT	XC-XA	XA-XC	YC-YA	YA-YC	GB-XB	WRITE, INC DA(YB)
6	IF GT	INC XA(XC)	XA-XC, XC-X	INC YA(YC)	YA-YC, YC-Y	XB-GB	READ(#), INC DA(YB)
7	GOTO	INC XA(XC+1)	XA-XC, XB=XC	INC YA(YC+1)	YA-YC, YB=YC	XB+GB	WRITE(#), INC DA(YB)
8	EXEC MACRO	INC XA(XC-1)	XC-X	INC YA(YC-1)	YC-Y	GB+0	READ, YB-DA
9	CALL	DEC XA(XC)	X-XC	DEC YA(YC)	Y-YC	GB-MR	WRITE, YB-DA
A	NA	DEC XA(XC-1)	XA-XC, XB-X	DEC YA(YC-1)	YA-YC, YB-Y	MR-GB	READ(#), YB-DA
B	DECJ, IF J#0	DEC XA(XC+1)	XA-XC, XB=X	DEC YA(YC+1)	YA-YC, YB=Y	MR+GB	WRITE(#), YB-DA
C	SEQ, DEC J	YB-XA	X-XC, XB=X	XB-YA	Y-YC, YB=Y	YB +GB	YB-DA
D	SEQ, YB-DEV	INC XA(YB)	XC-X, XB=XC	INC YA(XB)	YC-Y, YB=YC	GB-YB	YB=DA
E	SEQ, YB-J	DEC XA(YB)	XB-X, XB-XC	DEC YA(XB)	YB-Y, YB-YC	YB-GB	YB-DA(#)
F	RTN	CLR XA	XA-XC, XC-X, XB=XC	CLR YA	YA-YC, YC-Y, YB=YC	YB+GB	YB=DA(#)

CODE	XB	YB	TU	VW	F	H	M	IO
0	NOOP	NOOP	NOOP	NOOP	0	0	:22	INT HOST
1	VALUE	VALUE	F-T	H-W	T-ML	HB-HA	:P2	CLR Sbit
2	DX	DY	G-U	G-V	ML-T	HA-HB	:2P	CLR S
3	-	SCLV	F-T, G-U	H-W, G-V	T+ML	HA+HB	:PP	XB=IO
4	DCMV	R	XB-T, G-U	YB-W, G-V	T+0	HA XOR HB	:FR	XB=IO
5	U	U	XB-T	YB-W	T-GASN	HA OR HB	0	IOC
6	V	V	F-U	H-V	GASN-T	HA AND HB	B	ENABLE
7	T	W	XB-T, F-U	YB-W, H-V	T+GASN	HB+0	IO	DISABLE

FIGURE 3. CHI-5 MICRO OPERATIONS CHART (CONT)

CODE	TEST	RLD	GB	HA	HB	FCI	GCI	XL	YL	R	RA	HCI
0	SHIFT	NOOP	0	ML	U	NOOP	NOOP	NOOP	NOOP	RL	NOOP	NOOP
1	G	YB→RA	U	XB	U	GCO,FCI	HCO,GCI	XB→XL	YB→YL	RR	INC RA	LGCO
2	H	YB→RC	V	MR	V							
3	GH		T	YB	W							

M-FIELD REMAP

M-CODE	NEW FIELD	OPERATIONS
5	0	SEE D-FIELD
6	B	SEE HCI & RLD FIELDS
7	10	SEE IO-FIELD

U-MEMORY LAYOUT FOR MICRO-INSTRUCTIONS TO BE LOADED INTO PS-RAM

32b	
—	VALUE
PSA, TEST, R, RA, M, MULT	F, FG, G, GH, HA, H, XL, YL
XB, GB, TU, XA, X	YB, HB, W, YA, Y
16b	16b

CODE	MULT	CODE	MULT
0	NOOP	10	XL*MB
1	MA*MB	11	XL*XB
2	MA*XB	12	XL*YB
3	MA*YB	13	XL*XL
4	MA*XL	14	XL*YL
5	MA*YL	15	R*MB
6	XB*MB	16	R*XB
7	XB*XB	17	R*YB
8	XB*YB	18	R*XL
9	XB*XL	19	R*YL
A	XB*YL	1A	SHIFT*MB
B	YB*MB	1B	SHIFT*XB
C	YB*XB	1C	SHIFT*XL
D	YB*YB	1D	SHIFT*YL
E	YB*XL		
F	YB*YL		

1.1. ARITHMETIC PROCESSING UNIT (APU)

The APU contains a multiplier with two input registers, two auxiliary input registers and two output registers. The unit includes three adders and four accumulators. In addition, the APU contains logic to handle shifting, scaling, floating point normalization, bit reversal, and test logic for two of the adders.

1.1.1. MULTIPLIER

The 16-bit x 16-bit pipelined multiplier with input registers MX and MY produces a 32-bit product stored in two registers called ML and MR. ML (Multiply Product Left) contains the 16 most significant bits of the product; MR (Multiply Product Right) contains the least significant 16 bits.

A multiply is initiated in each instruction in which at least one of the multiplier input registers is loaded. The product is available in ML and MR in the second instruction following the one in which the multiply is initiated. The examples below illustrate the pipelined nature of the multiplier and its timing.

Examples: Dot product of a vector of length n with itself. Assume for all three examples that UV has been initialized to 0, and XA points to the first component of the vector.

Example 1: Assume in addition that J contains n-1.

instr1	X * X	"Initiate multiply X(i) * X(i)
instr2	INC XA	"Update address
instr3	MLMR + UV -> UV,	"Partial Sum + X(i-1)*X(i-1)
	DEC J,	"decrement loop counter
	if J>0 GOTO instr1	"test: done?

The same computation can be accomplished in a two instruction loop as follows:

Example 2: Assume n in J, MX set to 0.

instr1	X * X,	"Initiate X(i)*X(i)
	INC XA	"Update address
instr2	MLMR + U -> UV,	"Partial Sum + X(i-1)*X(i-1)
	DEC J,	"Decrement loop counter
	IF J>0 GOTO instr1	"Test: done?

1. The first time through the loop,
 - a) Instr1: $X(1) * X(1)$ is initiated, ($0 * \text{initial MY}$) \rightarrow MLMR
 - b) Instr2: $0 + (0 * \text{initial MY}) = 0 \rightarrow$ UV;
 J decremented to n-1.

These two instructions have 'filled the pipeline'.
Each additional time through the loop,

- a) instr1: $X(i) * X(i)$ is initiated, and
 $X(i-1) * X(i-1)$ is latched into ML, MR.
- b) instr2: $X(i-1) * X(i-1)$ is added to the partial sum.

The last time through the loop, an irrelevant multiplication is initiated. However, this product is not accumulated with the others, since J will decrement to 0 and cause control to pass out of the loop. The same computation can be performed in a loop of a single instruction.

Example 3. Assume J contains n+1, MX initialized to 0 at least two instructions earlier.

```
instr1  X * X,           "Initiate X(i)*X(i)
        INC XA,          "Update address
        MLMR + UV -> UV, "X(i-2)*X(i-2) + Partial Sum
        DEC J, GOTO instr1 "Test: done?
```

The first time this instruction is executed, $X(1) * X(1)$ is initiated and the 0 in MLMR is added to the 0 in UV. At the same time, $0 * \text{MY} = 0$ is latched into MLMR. XA is updated to point to the second component of the vector, and J is decremented to n.

The second time the instruction is executed, $X(2) * X(2)$ is initiated, and the (2nd) 0 in MLMR is added to the 0 in UV. During the same system clock period, $X(1) * X(1)$ is latched into MLMR. XA is updated to point to the 3rd component of the vector, and J is decremented to n-1.

These two initial executions of the instruction are required to fill the pipeline, and account for the n+1 loaded into J before the loop begins. In each succeeding execution of the instruction, a new multiply is initiated and a previous product is accumulated. In the last two executions of the instruction, irrelevant multiplies are initiated, but their corresponding products are never accumulated.

There are two auxiliary input registers to the multiplier called XL (X-latch) and YL (Y-latch). These are loaded from the X and Y bus respectively. XL can be input to either side of the multiplier; YL can act as input to the B-side only.

If MX is loaded in an instruction, but MY is not, the value used as the B-side multiplier in the previous multiply operation is used. A similar situation exists if MY is loaded in an instruction, but MX is not.

The mode of a multiplication is determined by a descriptor which indicates the nature of inputs and product. Figure 4 below shows the possible modes and their corresponding input and product types.

FIGURE 4
MULTIPLICATION OPERAND AND PRODUCT COMBINATIONS

MODE	OPERAND A	OPERAND B	PRODUCT
22	2's comp. integer	2's comp. integer	32-bit 2's comp. integer in ML, MR
P2	pos. binary integer	2's comp. integer	32-bit 2's comp. integer in ML, MR
2P	2's comp. integer	pos. binary integer	32-bit 2's comp. integer in ML, MR
PP	pos. binary integer	pos. binary integer	32-bit pos. binary integer in ML, MR
FR	2's comp. fraction	2's comp. fraction	16-bit 2's comp. fraction, convergent rounded in ML

Shifting is accomplished by multiplication by a power of 2.

For example, in the operation

$$\text{SHIFT } (P) * MY$$

where P is the value on the Y-bus, the actual multiplication performed is

$$2^{**Q} * MY$$

where Q is the low order 4 bits of P.

This operation shifts MY left Q places, or right 16-Q places, depending on the interpretation of results.

1.1.2. ADDERS

The CHI-5 provides three adders which can be used separately for 16-bit inputs, or in certain combinations for 32-bit or 48-bit inputs. The carries which provide the links between adders can be switched for single or double adder operations.

When a subtraction is performed which includes a carry from another adder, the quantity (1-carry) is subtracted from the minuend along with the subtrahend.

The F-adder is the most limited of the three adders, combining ML or the sign-extension of the A-input to the G-adder with T. (The sign-extension is 0 if the G-adder A-input is non-negative. It is -1, each bit is 1, when the G-adder A-input is negative). The F-adder output can be directed to either T or U.

Both the G and H-adders can combine three of the four accumulators plus zero on one side -- the B-side -- with either bus or either multiplier output register on the other -- the A-side. The operations performed in these adders are:

$$A+B, A-B, B-A, B+0$$

Each of these adders has an additional feature. The H-adder can perform the logical operations OR, XOR, AND. The G-adder can perform sums where one of the inputs is the absolute value of the quantity, Q, on the X or Y bus.

The output of the G-adder can be directed to U or V; the output of the H-adder can be directed to V or W. In addition, an operation can be performed in either adder and its output stored nowhere, but the result used to control program sequencing.

When an operation requires the 'FG adder', the F and G adders are linked, the carry-out from G (GCO) automatically being combined with the operation performed in F. In a 'GH' operation, the H carry-out (HCO) is automatically combined with the inputs to G.

In addition, GCO can be explicitly included in F-adder operations, HCO can be explicitly included in G-adder operations and LGCO (Last-G-Carry-Out -- the carry created by the G-adder operation of the previous instruction) can be explicitly included in H-adder operations. (Note: invocation of LGCO involves the remapping of the M field of the instruction, and conflicts with a multiplication, a D-memory operation or an IO operation).

1.1.3. ACCUMULATOR REGISTERS

The four registers T, U, V, and W are used primarily as accumulators in inner loops of array processes. Control of T and U is specified in the TU field; control of V and W is determined by the VW field.

The T register can be loaded from the F-adder or the X-bus; The W register can be loaded from the H-adder or the Y-bus. The U register is loaded from the F or G-adders, while the V register is loaded from the G or H-adders.

The output of the F-adder (resp H-adder) cannot be directed simultaneously to both the T and U registers (resp. the V and W registers).

For each of these registers, its value at the beginning of a system clock period can be used, and a new value stored, all within that clock period.

1.1.4. SCALING

The SCLV operator (SCALE of V) counts, for the contents of the V register, the number of consecutive leading bits (exclusive of the sign bit) which match the sign bit. This data is then gated onto the Y-bus. This operator is used, in conjunction with the SHIFT multiplication, in floating point normalization.

Note: The contents of V are assumed to be non-negative. If the contents of V are negative, SCLV does not produce a meaningful result. If V is zero, SCLV is 31. If the G-adder operation in the second preceding instruction caused an overflow, SCLV is 47. These special case values were designed to facilitate floating point arithmetic, and are recognized by the SHIFT test.

1.1.5. ARITHMETIC CONDITIONALS

Program sequencing can be controlled by the outcome of tests on the results of arithmetic operations. In all such operations, the branch is taken if the test condition is fulfilled.

The results of operations in the G, H or GH adders can be tested for being positive, negative, or equal to zero. Because of timing considerations, these tests must be made in the second instruction following the instruction in which the adder operation occurred.

Example: X + U: G	"instruction 1
any legal instruction	"instruction 2
IF G)0 GOTO LOOP	"instruction 3

If the result of the operation in instruction 1 is positive, control transfers to the instruction labeled LOOP.

There are two arithmetical conditional tests which were designed to facilitate micro-programming of floating point arithmetic. These tests divide the spectrum of Y-bus values into three sets. The tests are:

- 1) IF SHIFT SMALL GOTO psa
- 2) IF SHIFT OVFL GOTO psa

Both of these operations test the value on the Y-bus at a particular time. The first tests the Y-bus value of the previous instruction; the second tests the value on the Y-bus during the instruction executed just prior to the previous instruction. Thus both tests can be performed on the same Y-bus quantity.

SHIFT SMALL is true if the Y-bus value in the preceding instruction is between -16 and +15. SHIFT OVFL is true if the Y-bus value in the second preceding instruction is less than -32, or bit 5 is 1. Note, 47 satisfies SHIFT OVFL and 31 satisfies neither test; both would specify a right shift of one place, or a left shift of 15 places, if used in a shift multiply.

1.1.6. BIT REVERSAL

The DCMV operator (DECIMATION OF V) provides a bit reversed version of the contents of the V register. This reversed data can be gated onto the X-bus.

1.2. DATA OPERATIONS AND CONTROL

1.2.1. ARRAY MEMORIES

Two random access array memories -- X and Y -- provide large scratch memory for the arithmetic processing unit, as well as buffers for data to be processed. These 1024 word memories are connected, via the X and Y buses respectively, to the APU. X can be read or written (but not both) in every system clock period. The same holds true for Y.

The array memory address controllers -- XA and YA -- are identical. Associated with each is an auxiliary register -- XC (resp. YC) -- to help provide the flexibility necessary for computing along rows or columns of 2-dimensional arrays. Since the workings of the two controllers are identical, only operations involving XA will be described.

In any operation, XA can be incremented, decremented, cleared (set to zero) or loaded. The increment (resp. decrement) can be 1, 2, the contents of the XC register, XC+1, XC-1, or any quantity which can be gated onto the Y-bus. XA can be loaded, by a direct path, from the XC register, or it can be loaded by the value on the Y-bus.

A 32-bit wide, read-only memory -- R -- is provided to hold commonly used constants and coefficients (e.g. roots of unity). The R-memory is 1K long. The address -- RA -- from which table data is to be read can be set from the Y-bus. Similarly, the address increment -- RC -- is set using the Y-bus. RA and RC are registers in the R-memory address controller. The operation INC RA causes the value in RA to be incremented by the quantity in RC.

Either the left or right half of the currently addressed word of R can be gated onto the Y-bus or transferred directly to the A-side of the multiplier. The left 16 bits are referred to as RL (R-Left), the right 16 bits as RR (R-Right).

For each of these array memories, the address can be changed in one instruction, and data accessed at the new address in the next.

1.2.2. D-MEMORY

The D memory controller is organized around the 16-bit wide by 16 deep file register, whose output is an address for D memory access. The file element assignments are shown in Figure 5. When the M code is 5, the D field is available for D memory controller micro-instructions, as shown in figure 3.

The increment values in D Codes 0 to 3 are +1 for one 16-bit word read/write and +2 for two 16-bit words read/write. This access size is controlled by the most significant bit of the DA index. External DMA data transfers via the D-in/D-out lines can only be single 16-bit words. Thus, the CHI-5 can make

FIGURE 5
FILE REGISTER ASSIGNMENTS

LOCATION	ACCESS	ASSIGNMENT
0	CHI-5, HOST	(UNASSIGNED)
1	CHI-5, HOST	(UNASSIGNED)
2	CHI-5, HOST	LPC PARAMETER OUTPUT
3	CHI-5, HOST	(UNASSIGNED)
4	CHI-5, HOST	LPC PARAMETER INPUT
5	CHI-5, HOST	(UNASSIGNED)
6	CHI-5, HOST	(UNASSIGNED)
7	CHI-5, HOST	COMMAND PORT
8	CHI-5	MACRO LEVEL STACK POINTER
9	CHI-5	MACRO INSTRUCTION PROGRAM COUNTER
10	CHI-5	(UNASSIGNED)
11	CHI-5	(UNASSIGNED)
12	CHI-5	(UNASSIGNED)
13	CHI-5	(UNASSIGNED)
14	CHI-5	(UNASSIGNED)
15	CHI-5	(UNASSIGNED)

parallel two word transfers to D memory from the XB and YB or from D memory to the DX and DY registers. Logic in the D memory controller uses the D address least significant bit (DA(0)) to select DL or DR memory for one word data transfers.

D codes 4 to 7 use the value on the Y bus (YB) as a 16-bit two's complement increment. This mode allows the program to perform scattered address data transfers with the D memories. D Codes 8 to B (hexadecimal) allow the file register value to be changed to point to a new buffer while transferring the last data value in the old buffer. This accommodates the rotation of buffers without loss of extra cycles in the process.

1.2.3. I/O

The "IO" micro-instructions are available for use when the M code is 7. The 'IO' micro-instruction field operations are shown in Figure 6. The majority of the IO operations involve interrupt control. Ten interrupts can be set into the S register. All interrupts can be cleared by op code 2, or interrupts can be individually cleared by op code 1. The interrupt to be cleared is derived from the device register value. Interrupts 0 through 7 correspond to ports 0 thru 7 of the DA file, and are set when a Host transfer is complete. Interrupt 8 is used by the A/D and D/A FIFOs, to indicate a "half-full" condition. The host interrupt is used to post an interrupt in the host's DMA controller. IOC is presently an unused IO operation. Interrupt 9 is associated with the serial interface unit.

1.2.4. HOST INTERFACE

The Host interface is a 16-bit parallel input and output port that uses direct memory access (DMA) to the D memories. The host initiates a data transfer by providing a 3-bit host address, direction control line, data, if a host to D memory transfer is desired, and a data transfer request. The host address is used to read one of the first eight address registers of the 16 register file (DA file). The value in the DA file is used as the D memory address. The DL or DR memory is selected based on the address being even or odd, respectively. The interface controller issues an acknowledge when the word has been read or written. The file register is incremented by one. The host takes the data, if it is a D memory to host transfer, and requests the next word. When the host signals that the transfer is complete an interrupt bit is set in one of 8 interrupt register (S Reg) status bits. The interrupt denotes completion of a block transfer for a particular address

FIGURE 6
IO FIELD OPERATIONS

OP	DESCRIPTION
INT HOST	Set status bits in Command and Status register (CSR) of Host Interface equal to the value in register (port to be used). Set the attention bit in the CSR to interrupt the Host.
CLR SBIT	Set the S-bit corresponding to the currently selected device to 0.
CLR S	Set S register = 0
XB=IO	Gate output of currently selected device onto the X-bus. Interpretation of data depends on particular device. (Meaningful only if DEV>7).
XB->IO	Load the selected IO device with the value on the X-bus. Interpretation depends on device. (Meaningful only if DEV>7).
IOC	Device dependant operation for devices where DEV>7. Not currently implemented.
ENABLE	Enable interrupts.
DISABLE	Disable interrupts.

(port). In response to the interrupt, the CHI-5 program may re-initialize the port address file element in preparation for the next data block, and process the previously transferred block of data.

There is one port designated as the "Command-Port". It is used to block load a segment of executable macro-code into the D memories. At the completion of this load process, the command port interrupt to the CHI-5 causes the CHI-5 to execute the program.

1.2.5. VOICE TERMINAL OPERATION

The CHI-5 provides for voice terminal operation using an LPC-10 speech algorithm. The specialized I/O for the speech application includes analog-to-digital and digital-to-analog conversion. To accomplish these conversions, 8-bit mu-255-law CODEC chips are used.

The CODEC chips sample the analog signal at an 8 KHz sample rate and outputs digital data in a synchronous serial data stream. A 64 deep x 8-bit wide First In First Out (FIFO) memory part reconstructs the serial data into 8-bit parallel bytes. The FIFO buffers the data until the array processor is ready to take a block of data. The FIFO uses a "half full" interrupt to signal the processor that a block of 32 bytes is ready. The array processor will recognize the interrupt at a convenient point in the operation, and it will read the data out of the FIFO by programmed I/O onto the X bus.

A similar output operation exists. The array processor receives a "half empty" interrupt from an output FIFO whenever the FIFO has less than 32 bytes left. The data in the FIFO is loaded by the array processor by programmed I/O from the X bus. The data in the FIFO is shifted out as a synchronous serial data stream into the output section of the CODEC chips. The CODEC chips reconstruct an analog signal from the digital data.

1.3. PROGRAM CONTROL

The CHI-5 has a powerful program controller that can perform sequential program execution, unconditional branches, conditional branches, loop control, and subroutine nesting. The controller provides the unique capability of performing the loading of P memory RAM while still executing micro-code from P memory ROM. The EXEC MACRO micro-instruction gives the program controller the capability to decode the contents of the DL bus into two micro-code addresses, a 5-bit MAC1 address and an 11-bit MAC2 address. The 5-bit address specifies the first micro-instruction address of the MACRO and the 11-bit address specifies the second microinstruction address of the MACRO. After the second instruction, the normal program controller mode is restored.

Figure 7 gives the detailed op codes of the program controller micro-code.

FIGURE 7
PROGRAM CONTROL OPERATIONS

OP	DESCRIPTION
SEQ	PSA+1-→PSA. While executing the present micro-instruction, fetch its successor. (Default operation for this field).
IF STAT=0 GOTO val	If the status of the device presently selected is 0, branch to the address in the value field, otherwise sequence as above.
IF STAT=1 GOTO val	If the status of the device presently selected is 1, branch to the address in the value field, otherwise sequence as above.
IF TEST {=,(<,>) } 0 GOTO val	Arithmetic conditionals on the register selected by the TEST field. Branch when condition is met.
GOTO val	val-→PSA. Unconditional branch
EXEC MACRO	This instruction must be immediately preceded by the D-operation READ(EA). In this case, EXEC MACRO causes the selected argument gatherer to be activated and the micro operation to be accessed.
CALL val	PSA+1-→micro stack. val-→PSA. Increment the stack pointer. This is the micro-subroutine call.
DEC J	J-1 -> J. Sequence as for SEQ.
DEC J, IF J>0 GOTO val	If J>0, val-→PSA. Otherwise sequence. In either case J-1 -> J after test is made. Down counter loop control.
YB-→DEV	Select the device whose device number is on the Y-bus. Sequence as for SEQ.
YB-→J	Set J equal to the value on the Y-bus. Sequence as for SEQ.
RTN	Decrement the stack pointer. Micro stack -> PSA. Micro subroutine return.

The J loop counter is a special hardware register that enables the programmer to control the number of times a given microprogram loop is performed without using any APU registers or memory to keep the count. A stack memory is provided to hold the return address when a micro-code subroutine CALL is performed. The subroutine nesting can be 16 deep before the hardware stack memory overflows.

Program sequencing can be controlled by the outcome of tests on the results of arithmetic operations. In all such operations, the branch is taken if the test is fulfilled.

The results of operations in the G, H or GH adders can be tested for being positive, negative or equal to zero. The TEST field is used to select the adder whose operation is to be tested, while the P field determines the test to be performed. Because of timing considerations, these tests must be made in the second instruction following the instruction in which the adder operation occurred.

There are two arithmetic conditional tests which were designed to facilitate the micro-programming of floating point arithmetic. The tests are:

- 1) IF SHIFT SMALL GOTO psa
- 2) IF SHIFT OVFL GOTO psa

These tests are described in the section on the APU.

The STAT tests examine one of 16 status conditions as selected by the device register value. The first ten status conditions are the states of the ten S register bits used for interrupts. If the system interrupts are enabled, the controller recognizes an interrupt automatically during the EXEC MACRO instruction if any of the ten S bits are set. The recognition of an interrupt forces the controller to fetch the next micro-instruction from the fixed system interrupt address. This first level interrupt handler tests the S bits to find the interrupt cause. It then transfers control to the macro program designated for that S bit, after saving the address of the interrupted program.

Loading the program memory RAM is done by a special mode in the program controller in which the ROM addresses (PHA) and RAM addresses (PSA) are distinct. The program is transferred from D memory into RAM through the X and Y busses. Special bus transceivers connect the X and Y busses to the RAMs in the P memory. In addition, either RAM or ROM P memory can be transferred into the D memory, using the same data paths.

1.4. MACRO OPERATION

Programming the CHI-5 could be done completely in micro-code, but microcoding is very tedious work for applications programming. By microprogramming a library of common routines with generalized arguments, it is possible to build a large system program by a series of calls to this library. These "Calls" are termed "macro-code" because they are one level higher in sophistication than the micro-code. The CHI-5 supports this style of macro operation by adding a micro-code instruction called EXEC MACRO which tells the microprogram controller to decode the contents on the DL bus as a macro-instruction.

The 5-bit MAC1 field forces the program controller to address the first micro-instruction of the macro from the first 32 locations of P memory. The second instruction will be fetched from the 11-bit address specified in the MAC2 field. The first micro-instruction is used to fetch the first operand and initiate additional memory reads if needed. This allows the same micro-program to be used with several different sources for its first operand. The normal microprogram sequencer operation is restored after the second instruction of the macro. The number of macro arguments is determined by the programmer, however, the MAC1/MAC2 field must always be in the DL memory.

The macro library routines continue the MACRO mode by requiring the last 2 instructions of the macro to perform a READ of D memory via the Macro Instruction Program Counter (EA), followed by an EXEC MACRO instruction. The level of sophistication of a macro is entirely up to the program designer. Some macros may perform only housekeeping functions, while others may perform complex algorithms such as a Fourier transform, etc. Macros can be modularized into subroutines, subprograms, and even tasks. Therefore, the macro stack pointer is one of the pre-assigned file registers, along with the macro program counter and a host command port.

1.5. DATA FLOW

The CHI-5 uses many interconnecting data busses in order to accomplish a higher throughput processing rate. The multiple bus structure allows many parallel operations to be performed without interference. All of the data paths shown in Figure 1 are 16-bit unless otherwise noted. Many features of these busses are identical due to bilinear symmetry, thus many of the descriptions will be in pairs of busses.

The DL and DR busses connect the DL and DR memories to the DL/DR output registers, DX/DY registers, D-IN input port, XB/YB, and the program controller. The selection of DL or DR is

made by the DA address. Even addresses access DL and odd addresses access DR. The DX and DY registers are the input pipeline registers to the X and Y busses, respectively. During READ or WRITE operations, either 16-bit or 32-bit accesses can be made from the D memories by this configuration. DMA I/O operations are limited to 16-bit accesses only.

The 16-bit WRITE operation is restricted to the X BUS data for the source, but can write DL or DR depending on the DA address. The WRITE also does not have the pipeline delay as does the READ operation. During macro instruction fetch, the DL bus is connected to the program controller so that the MAC1 word can be used. Thus, macro programs must be aligned with MAC1 and MAC2 in DL memory and the first optional argument for the macro in DR memory.

The X and Y data busses connect the X and Y memories to the APU, DX and DY registers, DL and DR busses, VAL register, DEV register, J counter in the micro-code program controller, analog I/O FIFOs, and all of the P memory during micro-code load or store operations. The CHI-5 system X and Y busses dominate Figure 1. The X and Y busses are the major source of high speed data transfers to the APU. The X/Y memories can provide working storage of 1K words for each memory for the APU. The X and Y busses connect to the D memories as a path for moving blocks of data to and from the X/Y memories or APU. The X bus can be used to move data to both DL and DR memory, but the Y bus can be used to move data only to the DR memory.

Other than these main data paths, there are a number of system functions that connect to the X/Y busses. The VAL register can be read on either X or Y bus. The DEV register can be loaded only by the Y bus. The J counter in the micro-code program controller is loaded only by the Y bus. The D-memory controller uses the Y bus as one source for DA register data, for incrementing file data, and for loading file data. The X bus is used for reading and writing to the special analog I/O FIFOs and the serial interface unit.

A very special use of both X and Y busses is the reading and writing to the P memory. This special mode requires special bus transceivers to connect the X and Y busses to the P memory data busses. Also during this mode, the micro-code program controller is utilized such that the program being performed and the location being read or written have independent address sources (PHA and PSA). The ROM is cycled through a bootstrap routine that transfers micro-code data in the D-memories to the X and Y busses and writes the data into the PS RAMs or reads data from the P memories onto the X and Y busses and writes it into D memory.

2. CHI-5

The CHI-5 design is expressed in terms of a set of design modules which each provide a well defined subset of the total system. The physical realization of the modules required some adaptations of the ideal structure in order to fit the design onto the selected circuit boards.

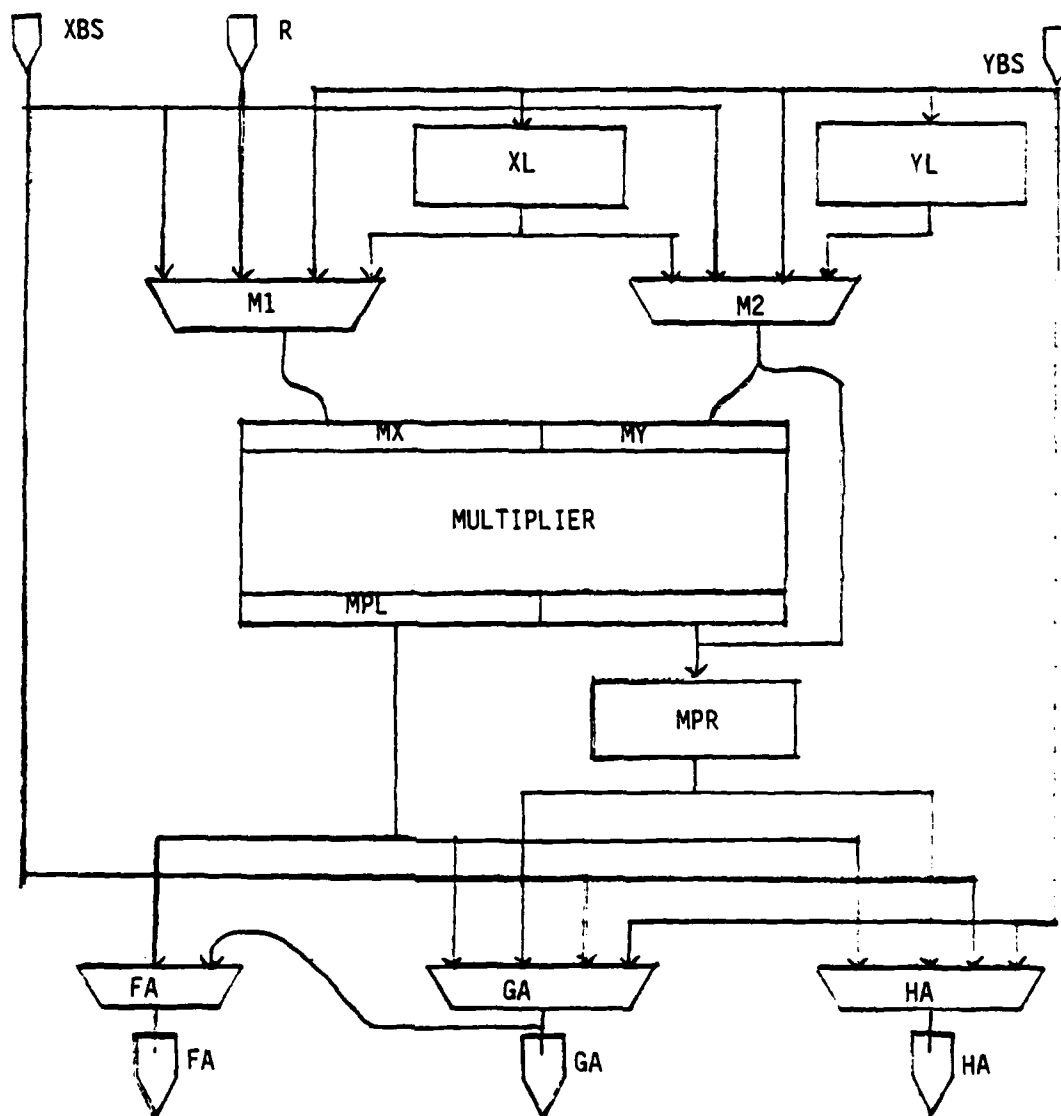
The board selected has space for approximately fifty integrated circuits and a primary connector with 116 contacts in addition to power and ground. The backplane motherboard has connections for slots for thirteen of these cards. The minimum CHI-5 configuration uses eleven cards; the remaining two slots are wired for additional D memory. In addition, the analog parts of the speech quality analog system are located on a printed circuit board which is isolated from the rest of the CHI-5.

In the remainder of this section, the ten distinct modules of the CHI-5 are described. In many cases, these descriptions will parallel the presentations in section one, with greater detail presented on the implementation. Other modules primarily provide the control, or "glue" which allows the system to function as a unit. These modules include the MULTIPLIER, ADDER1, ADDER2, and ACU, which together provide the major arithmetic and logic unit. The two AMEM modules, XMEM and YMEM, contain the array memory elements and much of the program memory. The PCU and STF modules provide the program sequencing and timing control for entire machine. The DCU, HIF1 and DMEM modules contain the main data memory and its addressing from either the CHI-5 or the host processor.

2.1. MULTIPLIER MODULE (MULT)

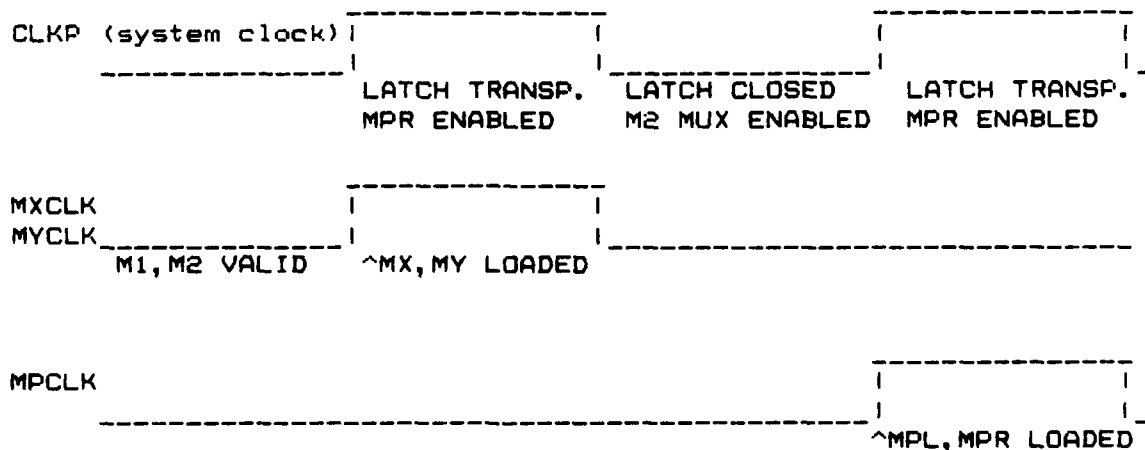
The multiplier module contains an LSI 16 x 16 multiplier chip with four to one input multiplexors for both inputs. It includes XL and YL, registers which can be loaded from their respective busses and which are direct inputs to the multiplier. It also contains multiplexors which select the input to one side of each of the three adders of the adder/accumulator module.

Figure 8. MULTIPLIER MODULE



Multiplies can be started using the contents of XBS, XL, YL, YBS, R, MX, and MY, in most, but not all, combinations. The input registers of the multiplier are loaded in one instruction. The output registers are automatically clocked to hold the product (MP) at the end of the following instruction time. The product is then available for input to the F, G, and H adders in the second instruction after the multiply was initiated. Because the MY inputs and the least significant bits of MP use the same pins, an external transparent latch is used to hold the least significant product bits. This latch is transparent during the first half of each clock period, when MPR is enabled out of the multiplier chip. The latch is closed at mid clock, holding the MPR value while the M2 input multiplexor is enabled and MPR output from the chip is disabled.

Figure 9. Multiplier Timing



An add using MPR requires a delay of the tri-state buffer in the chip, plus the throughput time of the latch. This delay is less than the access time of the array memories. There is sufficient time to complete a 48 bit add using the sign extended multiplier product.

In addition to the multiplier functions, the inputs to the "A" sides of all three adders are selected by multiplexors on the multiplier module. By locating these multiplexors here, three 16 bit parallel paths can be replaced by one or two 16 bit inputs on each of the adder units.

All controls for the multiplier are provided from the arithmetic control unit (ACU). Switched clocks for the multiplier input and output registers, XL and YL are generated on this module.

Due to the tightness of pinouts on this module, a cable is used to carry ten of the control signals from the ACU.

Figure 10. SIGNALS ON MULTIPLY MODULE CABLE CONNECTOR

<u>Signal</u>	<u>Function</u>	<u>Comment</u>
PRLI0	Sign control for MX	2's complement
PRLI1	Sign control for MY	or positive
RND	MP round control	Latched in chip
RS	MP shift control	One clock delayed
LXYR0,1	Select M1 output	
LXYL0,1	Select M2 output	
MXCPEN	Load MX from M1	
MYCPEN	Load MY from M2	

Figure 11. EXTERNAL SIGNALS FOR MULTIPLY MODULE

<u>Signal</u>	<u>Function</u>	<u>Comment</u>
XBS	Input to M1, M2, GA, GB, XL	Main data path
YBS	Input to M1, M2, GA, HA, YL	Main data path
R	Input to M1	
FA	Output of FA multiplexor	Input to F adder
GA	Output of GA multiplexor	Input to G adder
HA	Output of HA multiplexor	Input to H adder
FAZERO	Force FA to zero	From ACU
FMGASE	Select control for FA	From ACU
GAZERO	Force GA to zero	From ACU
GYRXL0,1	Select controls for GA	From ACU
HAZERO	Force HA to zero	From ACU
HYRXL0,1	Select controls for HA	From ACU
XLCPEN	Latch XBS into XL	From ACU
YLCPEN	Latch YBS into YL	From ACU
MPCPEN	Latch Multiplier Product	From ACU
CKP	System Clock Source	From STF

2.2. ADDER ACCUMULATOR MODULES

There are two adder accumulator modules in the CHI-5 because of the physical constraints of the circuit boards used. The ADDER1 module contains two 16 bit adders and three 16 bit accumulator registers. The remaining adder and register are located in the ADDER2 module, which also holds the ROM table memory.

The three adders all provide four basic operations: 0, A+B, A-B and B-A, where A and B are the two inputs. In addition, the G and H adders each provide specialized operations. The H adder can perform the binary logical operations AND, inclusive OR and exclusive OR. The G adder can compute the modified absolute value function

$$\begin{aligned}\text{MOD}(A) &= A && \text{if } A \geq 0 \\ &= -A-1 && \text{if } A < 0.\end{aligned}$$

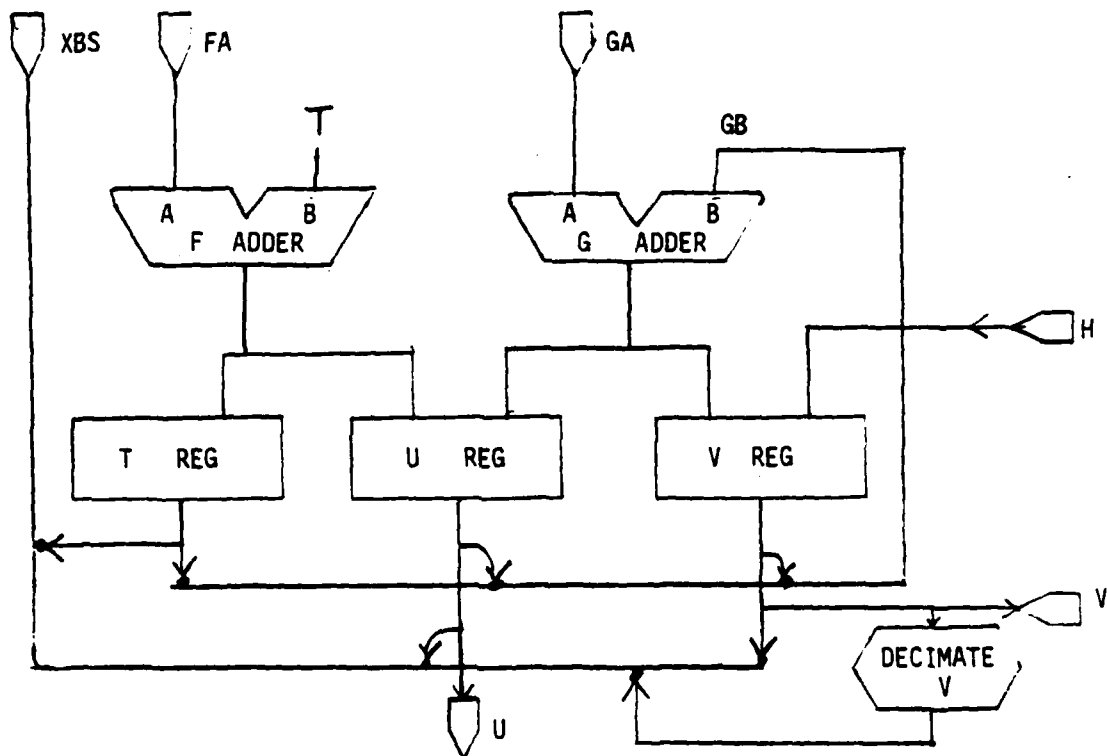
This function is of primary use in computing the number of places to shift a value in order to normalize it. It is used in conjunction with the scale logic on the STF module and the SHIFT decoder of ADDER2. The F adder is used for extended precision accumulation, since its A input can be selected to be the sign extension of the quantity on the A input to the G adder.

In addition to their independent operations, the adders can be combined to provide 32 or 48 bit accumulations in one clock. For higher precision operations, the carry out of the G adder is latched and can be used as the carry in to the H adder in the following instruction.

The four accumulator registers each work directly with a subset of the adders and data busses. T, U and V registers can drive XBS and be used as inputs on the B side of the G adder. U, V and W registers can drive the YBS and be used as inputs on the B side of the H adder. The output of the F adder, whose B input is always T, can be stored in T or U. The G adder output can be stored in U or V. The H adder output can be stored in V or W. This structure allows many combinations of adder operations and data transfers to take place in parallel, while keeping the micro-instruction decoding requirements simple and lending itself to straightforward implementation using standard components.

2.2.1. FIRST ADDER ACCUMULATOR MODULE (ADDER1)

Figure 12. ADDER1 CONTENTS



ADDER1 contains the F and G adders. It also includes the T, U, and V registers. The F adder's two inputs are FA and T. G adder has GA on one side and GBS on the other. GBS can be driven by T, U, or V.

Bus drivers on the ADDER1 module allow any one of the registers to drive the XBS. In addition, V can drive the XBS with its bits reversed end-for-end for use in FFT address calculations. Two status conditions GNEG and GZTST are computed on the ADDER1 module for use in arithmetic tests. The carry out of the G adder, GCO is used as the ripple carry input to the F adder when a 32 bit FG add is called for, and is latched for use as the carry into the H adder during the following clock period. Similarly, the carry out of the H adder, HCO, is an input to the module and can be used as the carry into the G adder for 32 bit GH adds.

Instruction decoding for the GBS source, XBS source, T inputs and U inputs is performed on the ADDER1 module. The remaining controls are prepared on the ACU module.

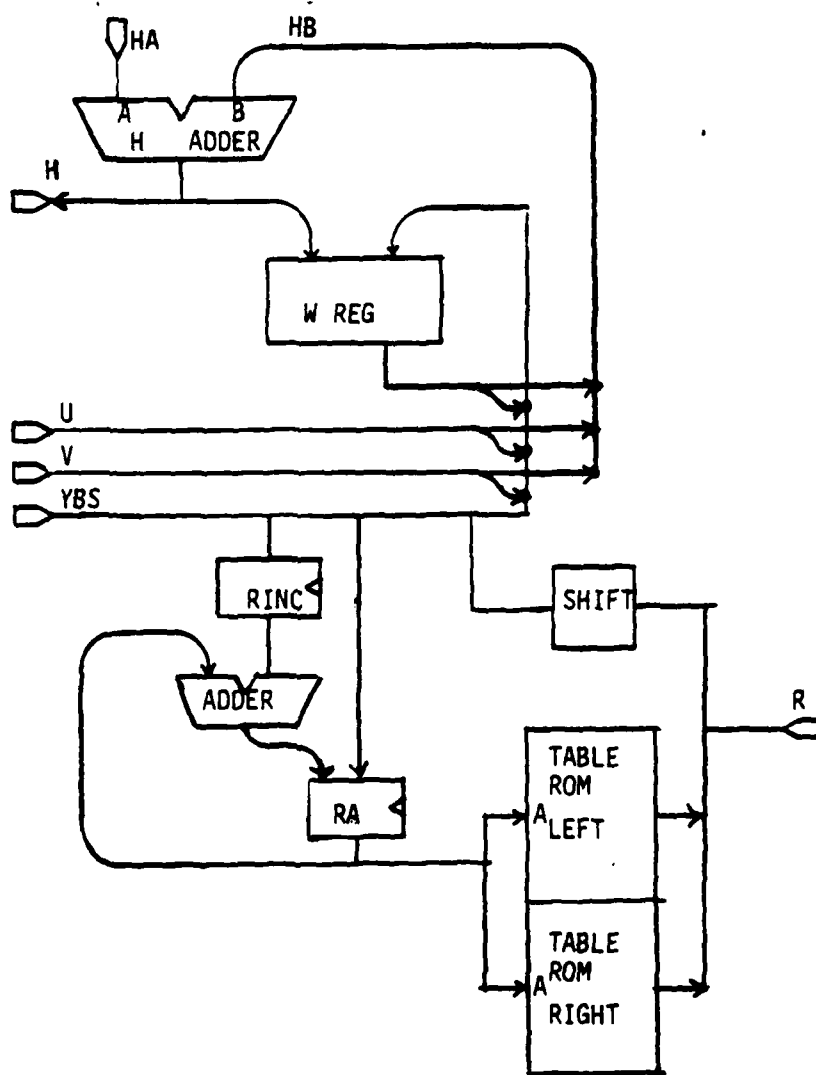
Figure 13. EXTERNAL SIGNALS FOR ADDER1 MODULE

<u>Signal</u>	<u>Function</u>	<u>Comment</u>
XBS	Input to T, Output of T,U,V	Main Data Path
FA	Input to F adder	From MULT
GA	Input to G adder	From MULT
H	Input for V register	From ADDER2
U	Output of U register	To ADDER2, STF
V	Output of V register	To ADDER2, STF
FOP0,1	F adder controls	From ACU
P63,64	GB select field	From XMEM
G0CI	Carry in to G adder	From ACU
G0P0I, G0P1I, G0P2I	G adder and FCI controls	From ACU
GCO	Carry out of G adder	To ADDER2, ACU
GZTSTL	G sum low order 15 bits = 0	To ACU
GNEG	G sum is negative	To ACU
P60,61,62	Control loading of T and U	From XMEM
HGSE	Select input for V	From ACU
VCLKEN	Load V register with input	From ACU
P65,66,67	Select XBS source	From XMEM
CKP	Pauseable system clock source	From STF

2.2.2. SECOND ADDER ACCUMULATOR MODULE (ADDER2)

The ADDER2 module contains the remaining third of the adder accumulator structure as well as the table ROM.

Figure 14. ADDER2 and ROM CONTENTS



**Best
Available
Copy**

2.2.2.1. H ADDER and W

The H adder takes HA and HBS as inputs. HBS may be driven by U, V, or W register, or it may be 0. Three status bits are generated by the H adder and sent to the ACU for use in arithmetic tests. These bits are H17, H2TSTL and HOVF. The H0CI input provides the input carry for the H adder.

The W register is loaded from H, or directly from the YBS. The bus drivers for gating U, V, W, or the ROM output onto YBS are located on this module.

2.2.2.2. TABLE ROM and SHIFT

The table memory ROM is 1024 words, 32 bits wide. It is split into two halves, left and right. One of the two halves is selected for access during each clock period. This structure allows switching back and forth between the two halves without updating the ROM address.

The ROM addressing structure includes an address register, an increment register, and an adder which combines them. The address register can be loaded from either the YBS or the output of the adder.

The ROM module also contains the SHIFT generator, a decoding circuit which provides a 16 bit power of 2 value, where the power is determined by the least significant four bits of the YBS. This value can be used instead of the table ROM to drive the R bus into the multiplier to produce a left shift of 0 to 15 places or a right shift of 1 to 16 places.

The sources for HBS and YBS, as well as the ROM left/right selection is determined by instruction bits which are decoded and buffered on the ADDER2 module. All other controls for the ADDER2 and ROM modules are prepared on the ACU.

Figure 15. EXTERNAL SIGNALS FOR ADDER2 AND ROM MODULES

<u>Signal</u>	<u>Function</u>	<u>Comment</u>
YBS	Input to W, Output of U,V,W	Main Data Path
U	Input to YBS and HBS	From ADDER1
V	Input to YBS and HBS	From ADDER1
HA	Input to H adder	From MULT
H	Output of H adder	To ADDER1, ACU
R	Output of ROM or SHIFT	To MULT
HCO	Carry out of H adder	To ADDER1
H2TSTL	H sum least 15 bits are 0	To ACU
HQVF	H sum produced an overflow	To ACU
P13,14	Select HBS source	From YMEM
H0CI	Carry in for H adder	From ACU
HQP0-2	Select H adder operation	From ACU
HYBSSE	Select input for W	From ACU
WCLKEN	Load W from its input	From ACU
P15-17	Select source for YBS	From YMEM
P111	Select ROM right	From XMEM
YBSESE	Select input for RA	From ACU
RACPEN	Load RA from its input	From ACU
RBCPEN	Load Inc Reg from YBS	From ACU
CKP	Pauseable system clock	From STF

2.3. ARITHMETIC CONTROL AND SERIAL INTERFACE MODULE (ACU)

The ACU module of the CHI-5 provides instruction decoding for the arithmetic modules ADDER1, ADDER2 and MULT. It also contains a separate section, the serial interface unit, which provides two independent ports for RS232C compatible devices.

2.3.1. ARITHMETIC CONTROL UNIT

The ACU contains the instruction buffers and decoding for the F, FG, G, GH, HA, H and VW fields of the microinstruction. These fields control the operation and A input selection for the three adders and the source selection for the V and W registers. The ACU generates the carry in for the G and H adders depending on the operation being performed on, if double adder operations are called for, the carry out received from the less significant adder.

All instruction buffering and decoding for the Multiplier Module is performed on the ACU. The MOP field is decoded and, if its value is 0-4, the M field is decoded to generate control signals for the multiplier. If the MOP is 5, 6 or 7, the clock enable signals for the multiplier inputs and the table ROM enable signal are forced low, preventing any multiply from starting and enabling the ROM instead of the SHIFT decoder. In any case, a one clock delayed logical OR of the multiplier input clock enables is generated to produce the clock enable for the multiplier product register. A decode of 4 in the MOP field cause the RND signal to be 0. The RND signal is one clock delayed and inverted for the RS control to the multiplier, since RS is not latched inside the multiplier chip. Several of the control signals to the multiplier are transmitted over a cable connector.

The controls for loading the table ROM address register and increment register, as well as the selection of the latched carry out of the G adder as the input to the H adder, are decoded from the M field only when the MOP is 6. In all other cases, they are forced low.

The ACU also buffers and decodes the XBS and YBS fields for sources not located in the ADDER or AMEM modules.

Finally, the status output from the G and H adders and the scale logic are received and latched. These status bits, together with the selected STAT line from the HIF module are selected for testing by the TEST and P field bits of the microinstruction to produce a signal CONL which is used by the PCU to determine whether or not a new instruction address is to be loaded. The TEST and P fields are not latched in an instruction buffer, so the new instruction address can be used to fetch the next microinstruction while the rest of the current

instruction is processed.

2.3.2. Serial Line Unit

The serial line unit consists of two Intel 8251A USART chips, a programmable baud rate generator, data and control registers for interfacing to the USARTs, bus drivers for the XBS and RS232-C line drivers and receivers. The unit recognizes device address 9 and uses the IOBL control to load its control and data registers. Bits in the control register address the USARTs and provide their read/write and mode controls. Control register bits also enable the data register onto the communication bus to the USARTs and baud rate generator and program the baud rate. The data output from an addressed USART is loaded onto the XBS when the IOAL control is received with device 9 addressed (IO1L = 0). The RS232C signals are available at the card edge of the ACU on a 20 pin connector.

Figure 16. EXTERNAL SIGNALS ON CABLE TO MULTIPLIER

<u>Signal</u>	<u>Function</u>
LXYR0, LXYS1	Input selection for MX
MXCPEN	Load MX from its inputs
LXYL0, LXYL1	Input selection for MY
MYCPEN	Load MY from its inputs
RS	right shift format control for MP
PRLI0, PRLI1	Sign Controls for MX and MY

Figure 17. EXTERNAL SIGNALS ON CABLE FOR SERIAL INTERFACE

<u>Signal</u>	<u>Function</u>
RXD0, RXD1	Input Serial Data from lines 0 and 1
DSR0, DSR1	Data Set Ready from lines 0 and 1
CTS0, CTS1	Clear to Send from lines 0 and 1
TXD0, TXD1	Output Serial Data to lines 0 and 1
RTS0, RTS1	Request to Send for lines 0 and 1
GND	Signal Ground for lines 0 and 1

Figure 18. EXTERNAL SIGNALS FOR ACU MODULE

<u>Signal</u>	<u>Function</u>	<u>Comment</u>
P(10-12), P(15-27), P(65-77), P(100-110), P(112-117)	Instruction bits from Program Memory	Source is XMEM or YMEM
GZTSTL, GNEG	G Adder status bits	from ADDER1
GCO	Carry out of G Adder	from ADDER1
HZTSTL, H17, HOVF	H Adder status bits	from ADDER2
HCO	Carry out of H adder	from ADDER2
SMD, BIGDL	Scale logic status	from STF
STAT	Selected status condition	from HIF1
IO1L	Serial Interface Selected	from HIF1
IOAL, IOBL	I/O transfer strobes	from STF
CKP	Pauseable system clock source	from STF
FOP0, FOP1	F Adder operation controls	to ADDER1
GOCI	G Adder carry in	to ADDER1
GOP(0, 1)I, GOPC	G Adder operation controls	to ADDER1
HGSE, VCPEN	V register input controls	to ADDER1
HOCI	H Adder carry in	to ADDER2
HOP0, HOP1, HOP2	H Adder operation controls	to ADDER2
HYBSSE, WCPEN	W register input controls	to ADDER2
YBESE, RACPEN	Table ROM address controls	to ADDER2
RBCPEN	Table ROM increment control	to ADDER2
ROMENL	Table ROM or SHIFT enable	to ADDER2
FAZERO, FMGASE	Controls for DA index	to MULT
GAZERO, GYRXL(0, 1)	Controls for GA multiplexor	to MULT
HAZERO, HYRXL(0, 1)	Controls for HA multiplexor	to MULT
MPCPEN	Clock enable for MP register	to MULT
XLCPEN, YLCPEN	Clock enables for XL and YL	to MULT
TVALXL	Enable VAL register onto XBS	to PCU
TVALYL	Enable VAL register onto YBS	to PCU
CONL	Load control for PSA	to PCU
MOPD7, MOPD5	I/O and D field selectors	to STF
TDLXL	Enable DX onto XBS	to DCU
TDRYL	Enable DY onto YBS	to DCU
OLDGNEG	Latched GNEG from G Adder	to STF
TSCLYL	Enable scale onto YBS	to STF
XB3QL	Unused XBS enable	
CMINT	Serial line interrupt signal	to HIF1

2.4. ARRAY MEMORY MODULE (AMEM or XMEM and YMEM)

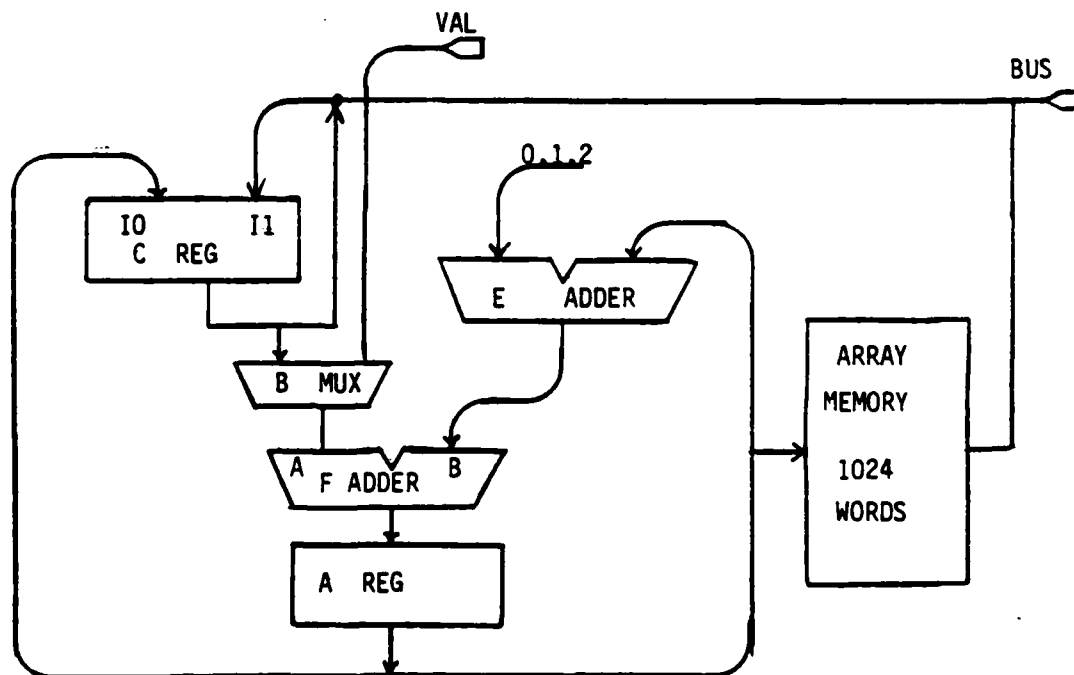
An array memory module contains a 1024 word by 16 bit data memory with its own addressing unit, plus 32 bits by 1024 RAM and 2048 ROM program memory. Two array memory modules exist in each CHI-5, XMEM and YMEM. These modules are identical.

The program memory is addressed and controlled from the PCU module. Of the 32 bits of the instruction word on these modules, 24 go to other modules and 8 are used to control the array data memory and its addressing.

2.4.1. ARRAY MEMORY

The array data memory is controlled by 15 control signals that are the outputs of two ROM decoders driven by the 8 bit instruction field. One of these signals, the memory write enable, must be sent to the STF for synchronization of memory write timing. The remaining controls are used directly within the module.

Figure 19. , ARRAY MEMORY and ADDRESSING



The array memory addressing structure includes an address register (A), an auxiliary register (C) and two adders (E and F). These adders are configured so that the E adder increments or decrements the current A register contents by 0, 1, or 2. It can also generate 0 as its output. The output of the E adder is then incremented or decremented in the F adder by the output of a two to one multiplexor (B), which selects either the contents of the C register, a value supplied from the opposite bus, or 0. The F adder output is used to update the A register at the end of the clock period. This structure permits addressing in two dimensions within the data memory.

The array memory can be driven onto its bus (XBS or YBS) or written from its bus each clock period. No latches are used in this path, the array memory data is valid on the bus within 85 nanoseconds of the beginning of the clock period. Data to be written into the array memory need not be valid on the bus until the last 65 nanoseconds of the clock period. Only one read or one write of each array memory can take place in each clock period.

2.4.2. PROGRAM MEMORY

The program memory is made up of 1024 words of writeable memory, called PS, and 2048 words of read only memory, called PH. The 32 bits of this memory located on each array memory module are divided into two 16 bit segments, PSL (PHL) and PSR (PHR). Transceivers are provided for loading PSL or PSR from the bus associated with the module, or for diagnostic reading of PSL, PHL, PSR, or PHR onto the bus. PS and PH are separately addressed and controlled to permit these program load and store operations. All program memory controls are prepared on the PCU and STF modules.

Figure 20. PROGRAM MEMORY

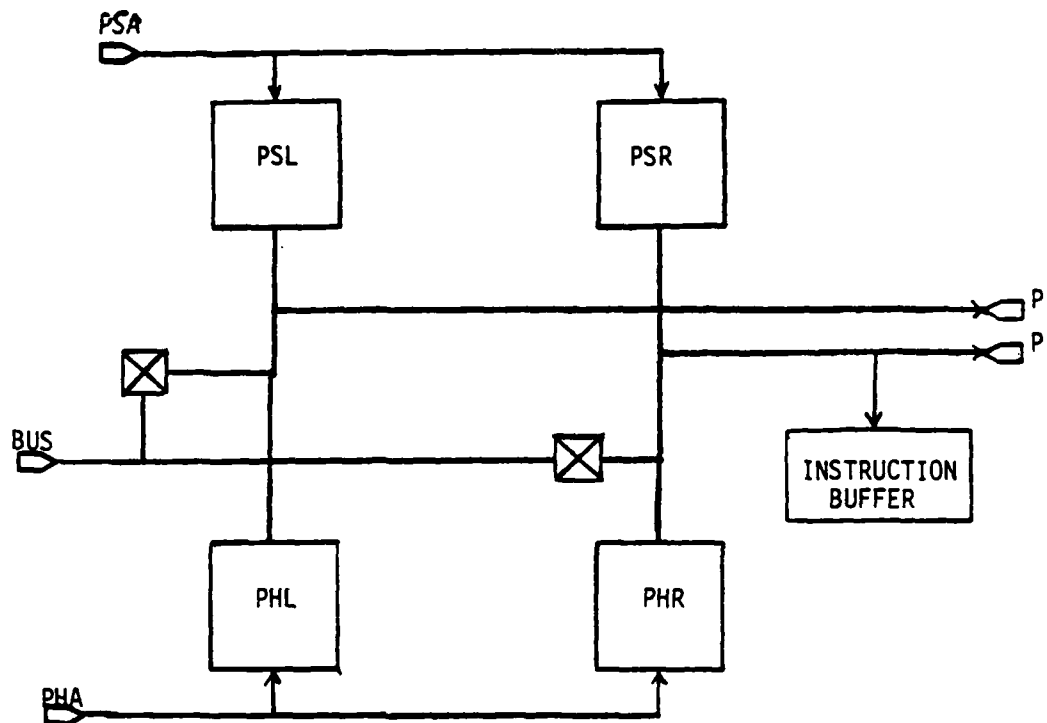


Figure 21. EXTERNAL SIGNALS FOR ARRAY MEMORY MODULES

<u>Internal</u>	<u>XMEN</u>	<u>YMEN</u>	<u>Function</u>
XBS	XBS	YBS	Data Bus to Data and Program Memories
PSA(00-11)	PSA(00-11)	PSA(00-11)	Program RAM Address
PHA(00-12)	PHA(00-12)	PHA(00-12)	Program ROM Address
PHPSE	PSEN	PSEN	Program RAM enable
PSPSE	PHEN	PHEN	Program ROM enable
XBSPREN	PSRWRTL	PSRWRTL	Write to PSR
XBSPLEN	PSLWRTL	PSLWRTL	Write to PSL
PSRLDL	PSRLDL	PSRLDL	Data Access to PSR, PHR
PSLLDL	PSLLDL	PSLLDL	Data Access to PSL, PHL
PSRD	PSRDQ	PSRDQ	Read PMEM onto BUS
CKP	CKP	CKP	System Clock Source
XWRTEN	XWRTEN	YWRTEN	Request to write AMEM
XWRTL	XWRTL	YWRTL	AMEM write pulse from STF
P(10-17)	P(60-67)	P(10-17)	Instruction word output
P(20-27)	P(100-107)	P(20-27)	
P(30-37)	P(110-117)	P(70-77)	

2.5. SCALE, TIMING and FIFO MODULE (STF)

This module contains the basic system clock circuits. It generates the narrow strobes used for memory write enables and data gating. It also contains several other functions which did not fit into the other modules.

2.5.1. Timing Functions

The central system clock is a four megahertz square wave. It and other clocks are produced from a 16 MHz crystal oscillator by two divisions by two. The first division produces an 8 MHz clock which is used to clock a register whose outputs are write pulses for D memory, X and Y array memories, and the three segments of RAM program memory. The program memory write pulses are active during the first half of system clock. The remaining memory write pulses are active during the second half of system clock. An early version of the system clock, called CK, is produced in the same way as the memory strobes by using the complement of the Q output as the D input. The paused system clock source CKP, is produced by another flip-flop using CK 'AND'ed with the active low PAUSL signal as the D input. The output, CKPL is then inverted with a power NAND gate to form the external signal CKP. CK is inverted, then inverted again in a power NAND gate to form the freerunning clock CLK.

The clock source CKP is used on each module requiring it to form a paused clock CLKP and switched clocks for registers by using CKP to clock a set of D flip-flops whose inputs are the enable signals for the registers, or true to make CLKP. The complement of CLKP is 'OR'ed with CKP to make an active low clear control to reset the derived clocks.

The system pause signal is produced under two circumstances. First, if a host memory access request (CRAQ) is present when the CHI-5 also requires a memory access, as indicated by either a decode of 5 in the M field (MOPDS) or an EXEC MACRO operation with a DOP of 0 or 1 (MR2DA), the PAUSQL latch is set. Second, during a program memory read operation the PRDL signal is active for one system clock period during each instruction. Either of these signals causes PAUSL to be active, which holds CKPL low and CKP high. When CKP is paused, no registers can load except for the DA register file and the host interface registers.

2.5.2. I/O and D Operation Decoding

This module also provides decoding and instruction buffering for the input/output and D memory operations. These operations are both decoded from the same bits as the multiplier control fields, their decoding is conditioned on the correct value in the M field.

I/O decoding is enabled by a seven in the M field and produces one of eight signals. These signals are latched after decoding to provide eight independent synchronous signals. Two of these, SETIEL and CLRIEL, are used to set or clear the interrupt enable signal, INTEN. INTEN and XMCR from the PCU are "AND"ed with the ANYS signal from the STF to produce the interrupt request INT.

D memory access decoding is enabled by a five in the M field. The instruction bits, together with the doubleword bit in the DA index register, and the host memory access signals CRBQ and H2AP, are combined to produce control signals for the DCU, DMEM and HIF1 modules.

2.5.3. Scale Logic

The Scale section contains a leading zero counter which is used to determine the number of places to shift a value in order to normalize it and test logic which sets status results to help control the type of shift required.

The leading zero counter is a 15 input priority encoder whose input is the contents of the V register, excluding the sign. Its output is a count from 0 to 14 of the number of leading zeros or 31 if V is 0. However, if the G adder output two instructions earlier produced an overflow, the count output is forced to 47. The count is enabled onto the YBUS by SCL2YENL.

The test logic examines the data on the YBUS to determine if the number of places to shift is less than 16 or if a very large shift or overflow is indicated. The two status conditions computed are BIGDL, which is active low if the YBUS value is less than -31 or positive and between 32 and 63, and SMD, which is true if the value is between -16 and +15.

2.5.4. Analog FIFO and Control

The FIFO section of the STF provides a buffered interface between the XBUS of the CHI-5 and the speech quality analog board. The analog board uses a codec and filter chip set to provide a converter between analog input/output and 8 bit mu-255 low digital values. A 2 MHz clock is used to drive the codec and filter. The analog board sends a serial bit stream and clock to the STF module. The serial signal is converted to an eight bit parallel value using a shift register and then loaded into a 64 word first-in/first-out memory. The load signal for the FIFO also clocks a five bit counter whose most significant bit output is AIQFL. When AIQFL changes from high to low, the S8 bit is set, interrupting the CHI-5. This provides an interrupt every 32 sample times. Similarly, eight bit values from a second FIFO are multiplexed onto the serial data output signal to the codec and filter.

The FIFO memories are emptied and filled, respectively, from the least significant eight bits of the XBS using the XBS → IO (IOAL) and XBS ← IO (IOBL) operations with device 8 selected.

Figure 22. EXTERNAL SIGNALS FOR STF MODULE

<u>Signal Name</u>	<u>Function</u>	<u>Comment</u>
CLK	Freerunning system clock	Source
CKP	Pauseable system clock source	
CACLK	Out of phase 8 MHz clock	to PCU
CRAQ	Host memory cycle request	from HIF1
PAUS	Active when system paused	to PCU
H2AP	Host transfer direction	from HIF1
CRBQ, CRBQL	Host memory cycle active	from HIF1
FAR4Q	Doubleword access bit	from DCU
DA00L	Least sig DMEM address bit	from DCU
MR2DA	Macro DOP is 0 or 1	from HIF1
MOPD5	Enable D field decoding	from ACU
P100-103	Instruction bits for D and I/O	from XMEM
BDAYL	Enable DA BUS onto YBUS	to DCU
MR2DAQL	Enable DY onto DA BUS	to DCU
FACLKEN	Load DA index register	to DCU
DCLKEN	Load DX and DY registers	to DCU
FWRTENL	Load DA file register	to DCU
Y2K	Select YBUS for DA increment	to DCU
K2FIN	Select adder for DA input	to DCU
KCIN	Carry in to DA address adder	to DCU
DBLE	Enable YBUS onto DRBUS	
HWRT	Enable host data onto DLBUS, DRBUS	to HIF1
APWRT	Enable XBS, YBUS onto DLBUS, DRBUS	to DCU
DWRTL	Disable data memory output	to DMEM
DLWRT	Write strobe for even memory addresses	to DMEM
DRWRT	Write strobe for odd memory addresses	to DMEM

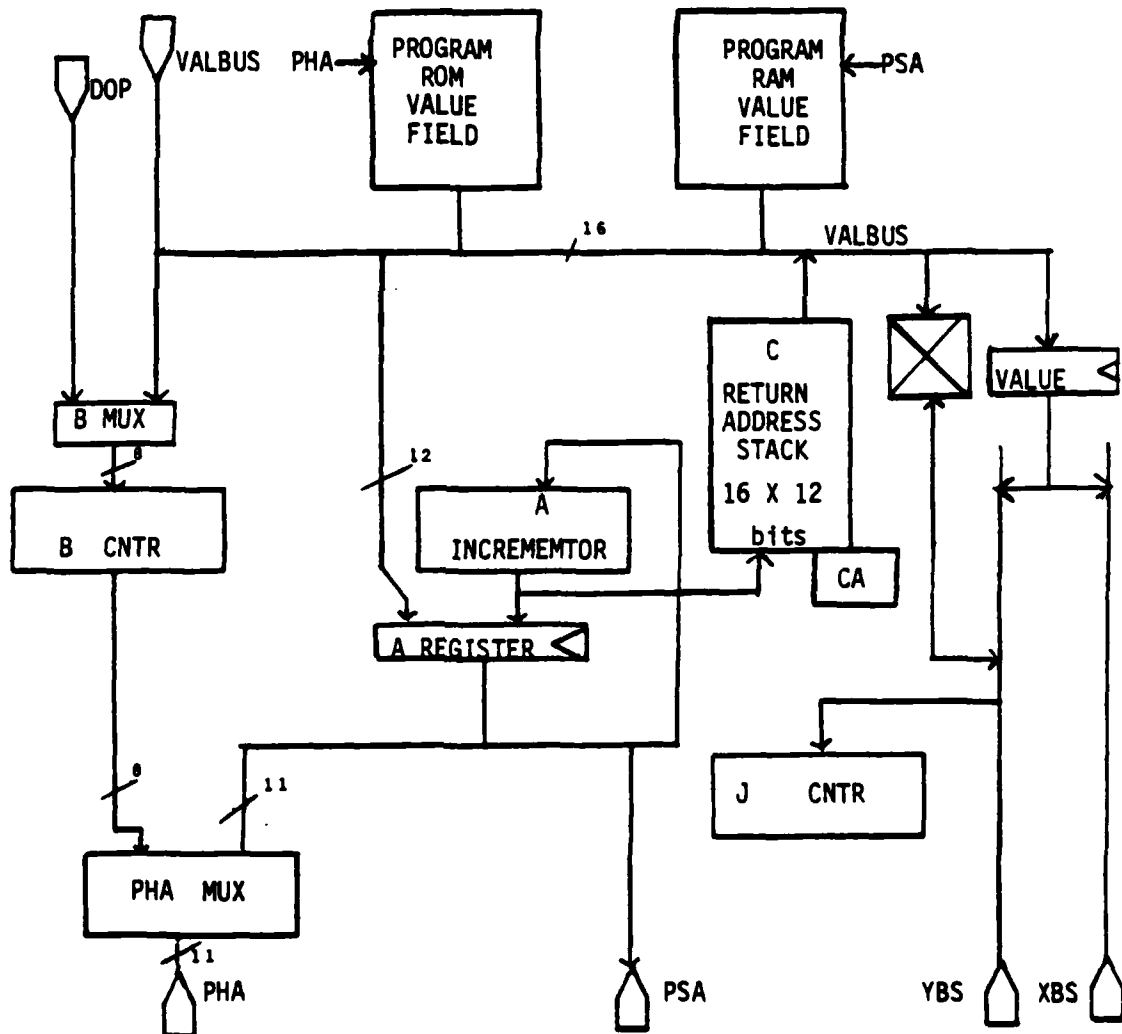
Figure 22 (cont.). EXTERNAL SIGNALS FOR STF MODULE

<u>Signal Name</u>	<u>Function</u>	<u>Comment</u>
XWRTE, YWRTE	Enable signals for AMEM	from XMEM, YMEM
XWRTL, YWRTL	Write strobes for AMEM	to XMEM, YMEM
PS0, PS1	Select segment of PMEM	from PCU
PSRD	Read PMEM	from PCU
PINH	Inhibit PMEM access	from PCU
PSRLDL, PSRWRTL	Load or store PMEM right	to XMEM, YMEM
PSLLDL, PSLWRTL	Load or store PMEM left	to XMEM, YMEM
PSVLDL, PSVWRTL	Load or store PMEM value	to PCU
PSRDQ	Read PMEM	to XMEM, YMEM, PCU
PLDL	Active during first half of PMEM load	to PCU
PRDL	Active during first clock of PMEM store	to PCU
MOPD7	Enable I/O decode	from ACU
IOAL	XBS to IO Device strobe	
IOBL	Gate IO data onto XBS	
IOCL	Unused IO strobe	
SETATNL	Sets host ATTN latch	to HIF1
CLRSIL, CLRSL	Clear S bits	to HIF1
ANYS	An S bit is set	from HIF1
XMCRL	EXEC MACRO Operation	from PCU
INT	Interrupt PCU sequencer	to PCU
V00-V16	Input to priority encoder	from ADDER1
U17, OLDGNEG	Used to determine overflow	from ADDER1, ACU
TSCLYL	Enable scale count onto YBUS	from ACU
SMD	YBUS value small	to ACU
BIGDL	YBUS value large (active low)	to ACU
ID0L	Device 8 selected	from HIF1
RESL	Reset, clears FIFO count	from HIF1
AIQFL	Sets S8 when low	to HIF1

2.6. PROGRAM CONTROL MODULE (PCU)

The PCU module contains the program sequencer for the CHI-5. It also contains the VALUE field, which is the last 16 bits of program memory, and an extra 4 bits of program memory (the extension bits) which are non-zero for only the first 256 locations in ROM address space. The value field of a micro-instruction is buffered in the VAL register on the PCU. This register can be used to drive either XBS or YBS. The value field is also used to specify the target address for a branch operation; the VALUE field of the instruction is one input to the program memory address register.

Figure 23. PROGRAM CONTROL MODULE



2.6.1. The Microprogram Sequencer

The program sequencer includes a twelve bit address register A whose output (PSA) is used to address the program memory RAM on the PCU and AMEM modules. PSA is also one input to a two to one multiplexor which determines PHA, which is used to address the program memory ROM on the PCU and AMEM. PHA12 is used to enable either RAM or ROM program memory. If it is 0, ROM is enabled, if it is 1, RAM is enabled. PSA is incremented by 1 in a dedicated adder whose output may be returned to A. The incremented PSA is also written into the current element of the 16 bit word file C.

When a branch operation is decoded in the ACU module a signal CONL is asserted to the PCU. This signal causes the input multiplexor to the A register to be switched to load it from the instruction value field, initiating a fetch at the new address. Meanwhile, the remaining fields of the instruction containing the branch are processed. If the branch is a subroutine call, the address counter for the C file is incremented, saving the address of the instruction following that containing the call. A subroutine return operation is decoded on the PCU module and causes the contents of the C file at the preceding address to be enabled instead of the value field of the program memory, and this value is selected for input to the A register.

The PCU also includes a 16 bit down counter, J which can be loaded from the YBS, decremented and tested for zero as part of a conditional branch instruction.

2.6.2. Macro Instruction Interpretation

In order to minimize or eliminate overhead in interpreting macro instructions stored in D memory, a special interpreting operation and hardware are provided in the CHI-5. The operation is called EXEC MACRO. When this operation is decoded in the PCU, the data currently on the DLBUS is interpreted as two micro program addresses. The upper 5 bits, called the DOP, are used to address one of the first 32 instructions in ROM program memory. The DOP is loaded into the auxiliary program address counter (B) through a two to one multiplexor and the B counter output is selected for the ROM program memory address, PHA. At the same time, the PCU sends a control signal (XMCRL) to the DCU, causing it to enable the remaining 11 bits of DLBUS onto the value field bus and this bus is selected for loading into the A register. As the rest of the instruction containing the EXEC MACRO operation is processed, the instruction selected by the DOP is fetched from program ROM. At the next clock, the PHA multiplexor is normally switched back to agree with PSA and the first instruction at the address loaded into A is fetched while the DOP micro instruction is processed. Thereafter, further instruction addresses are determined by the P field operation of the instructions processed.

2.6.3. Loading and Storing Program Memory

The auxiliary counter, B, can be used to address program ROM for more than one micro-instruction. This mode is used primarily for loading program RAM from D memory or for storing instructions from program memory in D memory. The B counter is used to address the programs used for loading and storing program memory; the A register holds the address of the instruction being loaded or stored. The four extension bits are used to enable this mode and control the reading and writing of program memory. The extension bits are LDMCR, which forces the PHA multiplexor to select the B counter for PHA, PSRD, which is a 1 to store program memory or a 0 to load it, and PS0 and PS1, which are decoded to select the portion of program memory to be operated on. If the value is 0, no operation takes place; if 1, the value field on the PCU module is loaded or stored using the YBS; if 2, program memory left on the two AMEM cards is loaded or stored using both XBS and YBS; if 3, program memory right is loaded or stored.

The load and store program routines are initiated as macro instructions with DOPs 37 and 36 respectively. The remaining 11 bits, loaded into the A register, are the starting program memory address to be loaded or stored. Each instruction in these routines except the last has the LDMCR extension bit set, forcing continued use of the B counter. The B counter is incremented by one each clock time unless loaded. When LDMCR is set, the A register is not loaded unless the PS0 and PS1 bits are both 1, so all three segments of a program memory location can be operated on in three separate instructions. To control the number of memory words to be transferred, the J counter of the PCU is used. A special P field operation is provided which loads the B counter from the value field bus if J is not zero.

Program memory can be loaded with no lost time for fetching the instructions of the load program. The instruction segment is written during the first half of the clock period, using transceivers on the AMEM and PCU modules to enable data from the XBS and YBS onto the memory I/O lines. At mid-clock, the transceivers are disabled and the program ROM is enabled, providing the next instruction in time for decoding at clock time. However, in order to be able to read both RAM and ROM program memory for diagnostic purposes, each instruction which reads a segment of program memory causes the normal system clock to be stretched, or paused, in the high state for an extra clock period. This allows the data being read to be present for a full clock period so that it can be stored in D memory. Then the PHA multiplexor can switch back to the B counter to fetch the next instruction for processing. All clock generation and memory write controls are implemented on the STF module.

2.6.4. Interrupt and Reset Processing

The PCU recognizes an interrupt signal, INT, at the time an EXEC MACRO operation is decoded. If active, INT causes the B counter to be loaded with a fixed address, 100, instead of the DOP value. This causes the first level interrupt routine to be started. This routine must immediately establish its own address by loading the A register using a branch instruction. The program address for the interrupted macro program is preserved by decrementing the macro instruction counter file cell to point at the interrupted instruction, then pushing it on the macro program stack as interrupt processing continues.

A RESET signal, RESL, forces the PHA multiplexor output to 400, initiating the reset microprogram. This program must also begin with a branch instruction to establish the proper instruction address in the A register.

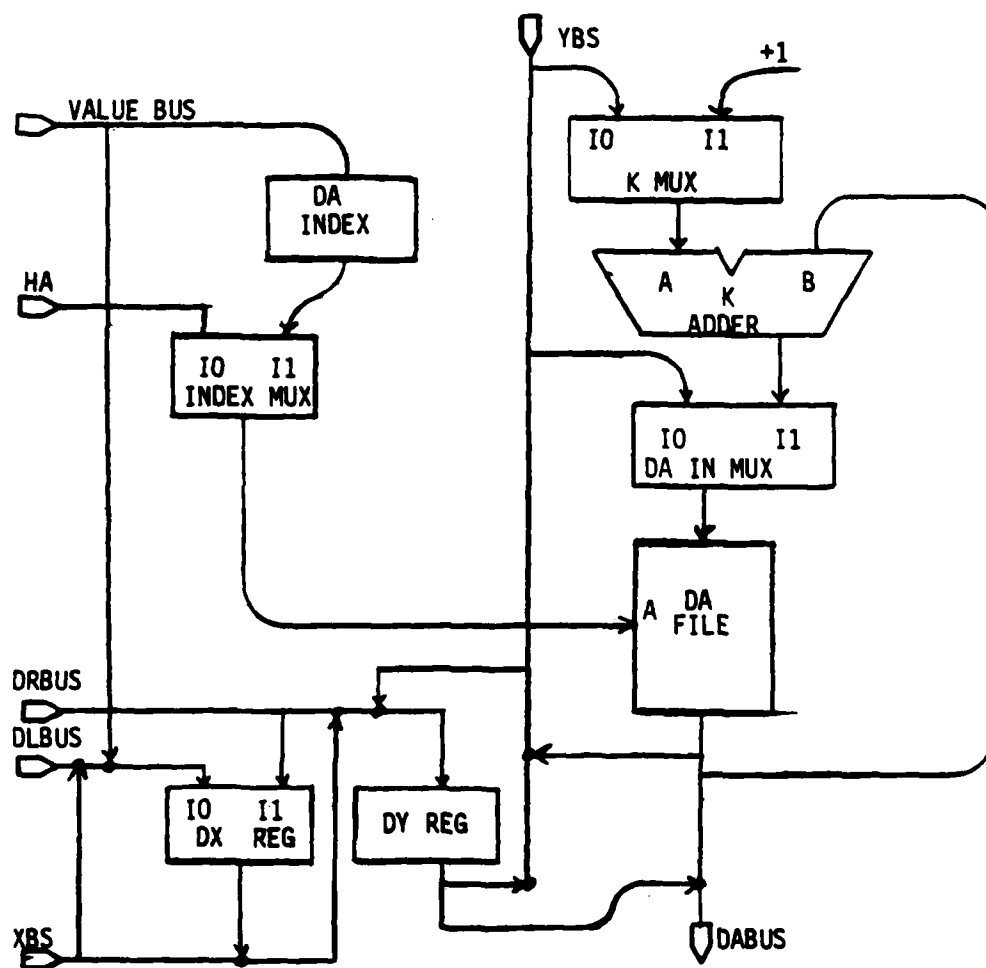
Figure 24. EXTERNAL SIGNALS FOR PCU MODULE

<u>Signal</u>	<u>Function</u>	<u>Comment</u>
P114-117	P field Instruction bits	From XMEM
CONL	Branch operation control	From ACU
CKP	System Clock Source	From STF
RESL	System Reset	From HIF1
PLDL	Enables Program Ram during first half of PS load op	From STF
PRDL	Selects PSA for PHA during first clock of PSRD op	From STF
INT	Interrupt request	From STF
PSVLDL	Enable data access to program memory value bits	From STF
PSRDQ	Gate program memory value bus to YBS	From STF
TVALXL	Enable VAL register onto XBS	From ACU
TVALYL	Enable VAL register onto YBS	From ACU
CACLK	Double frequency, out of phase clock for C File address counter	From STF
PAUS	Inhibit C File address counter	From STF
DOP(0-4)	Top 5 bits of DLBUS, gated into B counter by EXEC MACRO	From DCU
PS0, PS1	Program Memory Segment Selects	To STF
PSRD	Program Memory Access Direction	To STF
XMCR	EXEC MACRO signal	To STF and HIF1
DEVLDL	Load Device register from YBS	To HIF1
PHENL	Enable ROM program memory	To AMEM
PSENL	Enable RAM program memory	To AMEM
PINH	Active when PSA selected for PHA or during first clock of program memory read.	To STF

2.7. DATA MEMORY CONTROL MODULE (DCU)

The DCU module contains the addressing facility and data register for the main data memory. It supports from one to three data memory (DMEM) modules with up to 30K 32 bit doublewords of storage. The control and timing signals for the DCU and DMEM are provided primarily by the STF module. The HIF1 module interacts with the DCU to provide block transfers to DMEM from the host interface.

Figure 25. DCU MODULE



The addressing facility consists of a sixteen word file DA of address registers, an address adder K for incrementing the address register, a multiplexor KB which selects either 1 or the YBS as the increment quantity, and a multiplexor DAIN which selects either the output of the K adder or the YBS for input to the register file. The register file is addressed by a four bit index which is selected by a two to one multiplexor to be either the contents of the DA index register or a port address provided by the HIF1 module. The DA index register is loaded from the value field of the micro instruction; it also contains a fifth bit which selects single or doubleword mode for CHI-5 accesses to DMEM.

The output of the register file (DA) drives the DA Bus to provide memory addresses to the DMEM modules. The DA file output can be enabled onto the YBS for transfer to the CHI-5 ALU. It is also possible to disable the file output and enable the contents of the DY register onto the DA Bus. This path is used during EXEC MACRO operations with DOP=0 or 1 to fetch an initial parameter for a macro instruction before the macro processing begins.

The DCU provides data transfer gating and registers for CHI-5 processor access to DMEM. Data is read from DMEM in 32 bit doublewords over the DLBUS and DRBUS. The data on the DRBUS is loaded into the DY register. The DX register is loaded from the DLBUS if DA00 is 0 or from the DRBUS if DA00 is 1. DX can drive the XBS, DY can drive the YBS or the DABUS as described above. Bus drivers are provided on the DCU to allow data from the XBS to drive the DLBUS or DRBUS and for data from the YBS to drive the DRBUS for writing DMEM. For single word writes, XBS provides the data and is enabled onto the XBS or the YBS based on DA00. For doubleword writing into DMEM, XBS data is also enabled onto the DRBUS. The DCU also provides the DLBUS data to the PCU during an EXEC MACRO operation. The signal XMCRL is used to enable the least significant 11 bits, with a high order 0, onto the instruction value bus (BUS). The most significant 5 bits of DLBS are input directly to the PCU as DOP.

Figure 26. EXTERNAL SIGNALS FOR DCU MODULE

<u>Signal</u>	<u>Function</u>	<u>Comments</u>
XBS (00-17)	Primary path for data	
YBS (00-17)	Path for address information	
DLBUS (00-17)	D memory data for even addresses	
DRBUS (00-17)	D memory data for odd addresses	
DABUS (00-17)	D memory address	to DMEM
HA (0-2)	Port number to select DA register 0 to 7 for Host memory access.	from HIF1
BUS (00-13)	Instruction value bus	to PCU
DCLKEN	Load DX and DY from DLBUS, DRBUS	from STF
FACLKEN	Load DA index register from BUS	from STF
K2FIN	Selects K adder for input to file	from STF
Y2K	Selects YBS for increment in K adder	from STF
KCIN	Used to increment by 2 instead of 1	from STF
FWRTENL	Load DA file from its input	from STF
APWRT, DBLE	Enable XBS and YBS onto DLBS and DRBS	from STF
MR2DAQ	Enable DY onto DABUS	from STF
BDAYL	Enable DABUS onto YBUS	from STF
TDRYL	Enable DY onto YBUS	from ACU
TDLXL	Enable DX onto XBS	from ACU
CRBQ	Select HA to address DA file	from HI
FAR4Q	Doubleword access mode	to STF
CLK	Free running System Clock	

2.8. HOST INTERFACE MODULE (HIF1)

The host interface module provides a basic interface compatible with parallel DMA interfaces to commercial mini-computers. Data transfers with the host access DMEM to fetch or store 16 bit words using one of eight DA registers to address the memory.

The host controls the data transfer through a request/acknowledge handshaking protocol. When a request is received, it is synchronized with the system clock (CRAQ), then the memory transfer takes place during the following clock period (CRBQ). During this time, the host specified port (HA) is used by the DCU to select the file register to use for DA. If the request is a transfer into D memory, the host output data lines (DAT00-15) are enabled onto either DLBS or DRBS, depending on DA00. For data transfers from D memory to the host, the data from DLBS and DRBS is loaded into two 16 bit registers on the HIF1; at the same time, DA00 is latched. The appropriate register is then gated onto the host input data lines (DATI00-15) based on the latched DA00. When the host signals that the transfer is completed, the S register bit corresponding to the HA used is set.

The host can initialize the CHI-5 through either of two active high RESET lines. These are 'OR'ed to form the reset signal (RESL), which causes the CHI-5 to start its initialization microcode.

The CHI-5 signals the host using the HOST INT micro operation. This operation sets the ATTN latch and loads a three bit status word from the DEV register. These four bits are sent to the host. When the host acknowledges the ATTN by a low to high transition on the ATNCLR signal, the ATTN latch is cleared. The ATTN latch is also connected to the status multiplexor input 10 so the CHI-5 program can know when its interrupt has been recognized.

The S register includes eight bits associated with the eight host ports and two bits set by the analog and serial line interfaces, respectively. The "OR" of these ten bits is sent to the STF, where it is used to generate an interrupt during EXEC MACRO operations. The S register bits, as well as 6 other status conditions, are inputs to a sixteen to one multiplexor controlled by the DEV register to produce a STATUS signal which can be tested by the PCU. The S bit addressed by the DEV register is cleared by the CLRSIL signal from the STF or all ten bits can be cleared by the CLRSL signal.

The DEV register itself is loaded from the YBS by the DEVLDL signal from the PCU. In addition to the functions already described, the DEV register contents are decoded to produce

active low device select signals I00L to I07L for DEV selections of 8 to 15.

Figure 27. HOST CABLE SIGNALS

<u>Signal</u>	<u>Function</u>
DAT00-15	Output data from host
DATI00-15	Input data to host
HAP	Transfer is from host to CHI-5
BURST	Controls SCY output
RESET	Resets CHI-5
ATNCLR	Clears ATTN latch
EOC	Memory Access handshake from host
RDY	Cleared to start transfer, set to terminate it.
STAT0-2	Status bits from CHI-5 to host
ATTN	Signal from CHI-5
CREQA	Handshake acknowledge from CHI-5 to host
SCY	Used for "burst made" transfers
INIT	Resets CHI-5
C1	Direction control, inverted HAP returned to host

Figure 28. EXTERNAL SIGNAL NAMES FOR HIF1 MODULE

<u>Signal</u>	<u>Function</u>	<u>Comment</u>
DLBS00-17	D Memory Bus for even address	
DRBS00-17	D Memory Bus for odd address	
YBS00-03	Source for loading DEV register	
DEVLDL	Load control for DEV register	From PCU
DA00	Least significant memory address bit	From DCU
SETATNL	Set ATTN latch	From STF
CLRSIL	Clear selected S bit	From STF
CLRSL	Clear all S bits	From STF
CLK	Free running System Clock	From STF
CKP	Source for paused system clock	From STF
CMINT	Set S bit 9	From ACU
AIQFL	Set S bit 8	From STF
HWRT	Enable host data onto DLBS or DRBS	From STF
S11-15	Inputs to status multiplexor	Unused
XMCRL	Enable for MR2DA output	From PCU
MR2DA	DLBS17-14=0 during EXEC MACRO	To STF
ANYS	At least one of S0-S9 is set	To STF
STATUS	Selected S bit is set	To ACU
CRBQ,CRBQL	Active during host access to DMEM	
CRAQ	Host request for DMEM	
HA0-2	Host Port number	To DCU
RESL	System Reset Signal	
H2AP	Transfer is from host to CHI-5	To STF

2.9. DATA MEMORY MODULE

Each data memory module provides 8,192 32-bit double words of RAM storage and either 1024 or 2048 doublewords of PROM. The RAM chips are arranged in 4 rows of 2K words each.

All control, address and data signals for the DMEM module are buffered at the card.

The DA BUS address input is received through transparent latches which are closed at mid-clock to hold the address at which data is read or written over DLBUS and DRBUS. Each of three possible DMEM slots in the backplane motherboard is wired to recognize a different set of addresses for its RAM and ROM. The most significant two bits of DA are decoded and codes 0, 1 and 2 are wired as enable signals for the RAM on the three boards. The next two bits of DA provide chip select signals for the enabled RAM chips. If the board select decode is 3, the next two DA bits and their complements are decoded to enable the ROM chips on each board.

As the address inputs are held fixed at mid-clock, data output for a read is enabled from the selected row of chips and gated through transceivers onto the DL and DR busses, if the board is enabled. If the operation is a write into data memory, the transceivers gate data from the DL and DR busses onto the module and independent DLWRT and DRWRT signals from the STF are used to write either or both halves of the doubleword.

Figure 29. EXTERNAL SIGNALS FOR DMEM MODULE

<u>Signal</u>	<u>Function</u>
DLBUS (00-17)	Bidirectional data path for left 16 bit words
DRBUS (00-17)	Bidirectional data path for right 16 bit words
DABUS (01-17)	Address input from DCU
BDSL (0-3)	Decode of DA17 and DA16 output
BDSL	Input wired from BDSL0,1 or 2 to enable RAM
A13,A13L,A14,A14L	Complementary states of DA14 and DA15 output
RA13,RA14	Inputs ANDed with BDSL3 to enable ROM
DWRTL (OUTEN)	Output enable during second half of clock
DLWRT	Write pulse for left 16 bit word
DRWRT	Write pulse for right 16 bit word

GLOSSARY

- D Instruction field for DMEM operations.
- DA Collection of 16 cells which act as address registers for D-memory.
- DCMV Decimation of V register. (Bit reversal)
- DEV Device register.
- DMEM Main random access memory.
- DX A register into which data from DMEM is placed after a READ operation. (All single word READs or the high-order 16 bits of a double word READ).
- DY Register into which data from low-order 16 bits of a double word READ is placed.
- EA Macro instruction execution address (in D-memory)
- EXEC Execute (a MACRO).
- F F-adder.
- FA A-input to the F-adder.
- FCI F-adder carry-in.
- FG Linked F and G adders. Instruction field used to link F and G adders.
- FR Multiplication mode - Inputs and product in fractional form.
- G G-adder.

GA A-input to G-adder.

GB B-input to G-adder.

GCI G-adder carry in.

GCO G-adder carry out.

GH Linked G and H adders. Instruction field used to link G and H adders.

H H-adder.

HA A-input to H-adder.

HB B-input to H-adder.

HCI H-adder carry-in.

HCO H-adder carry-out.

IO Input/Output. Instruction field for input/output operations.

J Tally register.

LBCO Last G-adder carry-out (carry-out from the G-adder operation of the previous instruction.

M Instruction field for multiplication modes.

ML Multiplier product left. High-order 16 bits.

MR Multiplier product right. Low-order 16 bits.

MX A-input to multiplier.

MY B-input to multiplier.

P Instruction field for program control.

P2 Multiplication mode. Positive x 2's complement.

PP Multiplication mode. Positive x Positive.

2P Multiplication mode. 2's complement x Positive.

22 Multiplication mode. 2's comp. x 2's comp.

PHA ROM program source address multiplexer output.

PSA RAM program source address register.

PSB Auxiliary program source address register.

RMEM ROM table memory.

RA Register containing R address.

RC Register containing R address increment.

RL R left. High-order 16 bits of currently addressed word in R.

RR R right. Low order 16 bits of currently addressed word in R.

RLD Instruction field for loading of RA and/or RC.

S Status register.

SCLV The scale of the contents of the V register.

SHIFT Multiplier used to shift (normalize, align, scale) a number.

STAT Device condition which can be tested.

T Arithmetic register.

TEST Instruction field to select adder for conditional branch test.

U Arithmetic register.

V Arithmetic register.

VAL Register loaded from the VALUE field of the instruction.

val Non-negative constant used in an instruction as a DMEM address.

W Arithmetic register.

X Array memory.

XA X-memory address register.

XB X-bus.

XC Auxiliary X-memory address register.

XL X-latch. Auxiliary multiplier input register.

Y Array memory.

YA Y-memory address register.

YB Y-bus.

YC Auxiliary Y-memory address register.

YL Y-latch. Auxiliary multiplier input register.